SUPERNOVA

ScreenShot

COLLANA DI PROGRAMMAZIONE

COME PROGRAMMARE PASSO PER PASSO

COMMODORE 64



PHIL CORNES

LIBRO 1

Indispensabile per chi inizia finalmente la prima guida chiara e semplice per imparare a programmare - interamente a colori -

Screen Shot

COLLANA DI PROGRAMMAZIONE

COME PROGRAMMARE PASSO PER PASSO

COMMODORE 64

LA COLLANA DI PROGRAMMAZIONE SCREEN SHOT

Non c'è mai stato, come oggi, un bisogno così urgente di una serie di guide pratiche, facili, ben fatte, per imparare a usare il computer. La collana Screen Shot è stata concepita proprio per questo. È un concetto completamente nuovo nel campo dell'autoistruzione per i calcolatori. Ed è la prima collana di manuali dedicati a macchine specifiche, completamente illustrati, che insegnano a programmare passo per passo.

LIBRI SUL COMMODORE 64

Questo è il primo volume di una serie di guide, uniche nella loro concezione, che insegnano a programmare passo per passo il Commodore 64. Insieme con gli altri volumi, questo libro costituisce un corso di programmazione completo e autosufficiente, che inizia dai principi di base e prosegue, descrivendo programmi e tecniche via via sempre più sofisticati, fino a un livello avanzato.

NELLA STESSA COLLANA

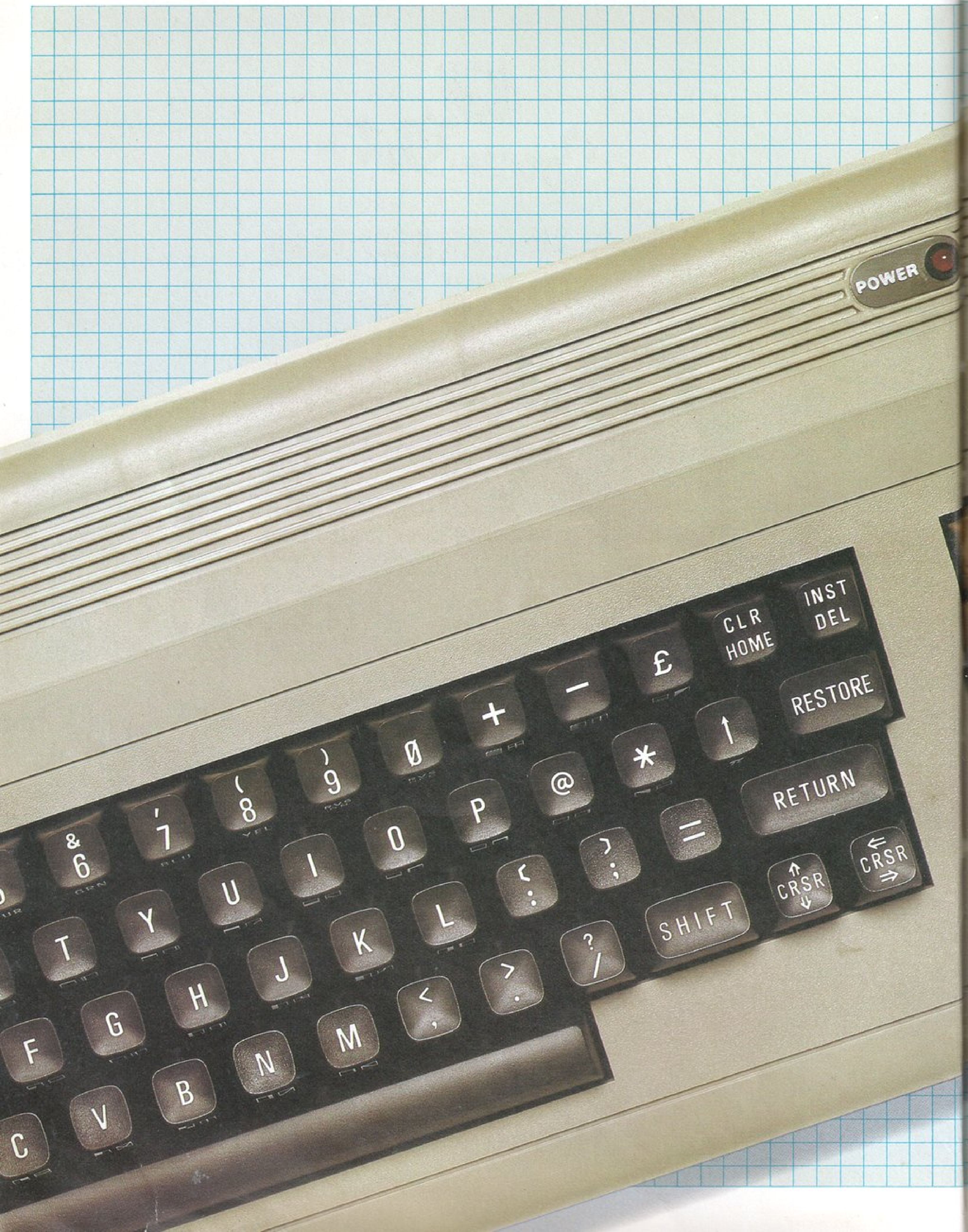
Come programmare passo per passo
ZX Spectrum e ZX Spectrum plus

PHIL CORNES

Dopo aver studiato matematica e programmazione, Phil Cornes ha lavorato alla realizzazione di sistemi educativi basati su computer al National Training College del British Telecom. Dal 1978 lavora anche come autore di pubblicazioni tecniche e collabora a varie riviste sui personal computer, come *Personal Computer World*, *Computing Today* e *Electronics Today International*. Ha scritto un libro e molti articoli sull'uso e sulla programmazione del Commodore 64.

SUPERNOVA

LIBRO 1

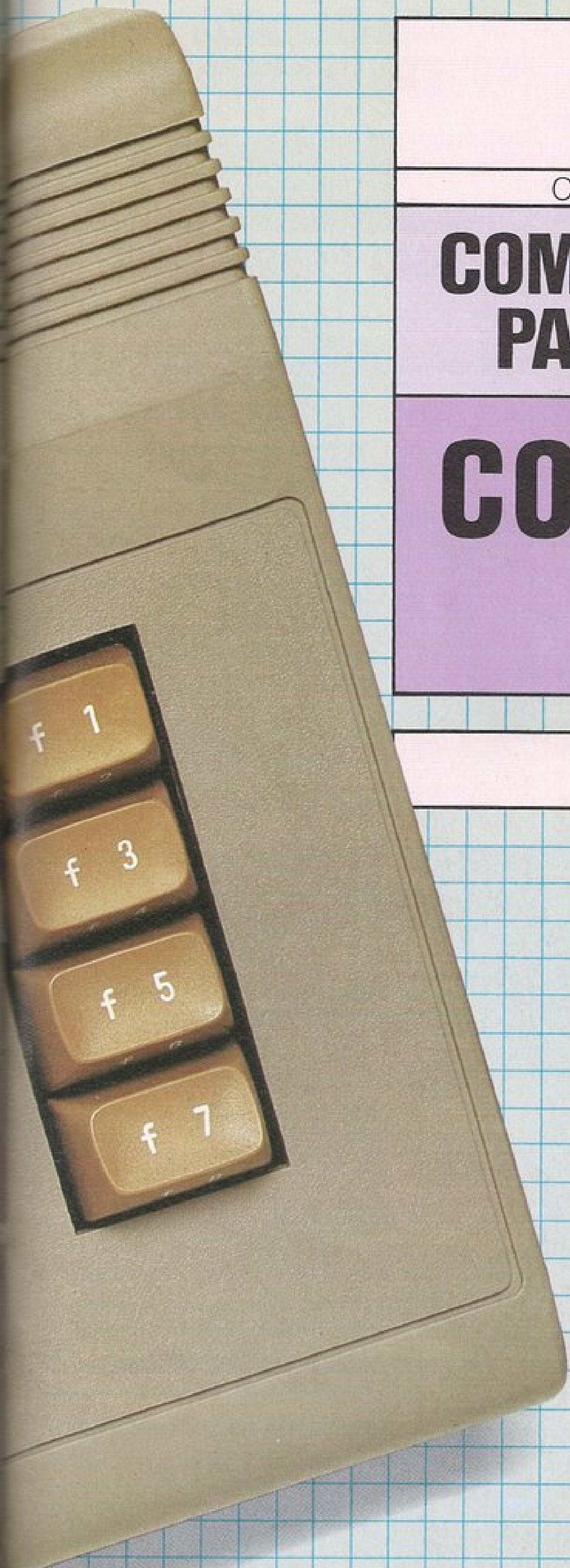

POWER
INST DEL
CLR HOME
RESTORE
RETURN
SHIFT
CRSR
CRSR

Screen Shot
COLLANA DI PROGRAMMAZIONE

COME PROGRAMMARE PASSO PER PASSO

COMMODORE 64

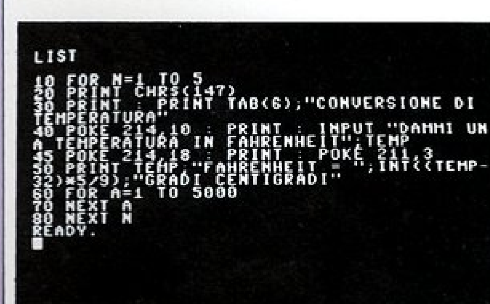
PHIL CORNES



SUPERNOVA

LIBRO 1

SOMMARIO



La collana Screen-Shot è stata ideata e disegnata da Dorling Kindersley Limited, 9 Henrietta Street, Covent Garden, London WC2E 8PS.

Designer Hugh Schermuly
Fotografie Vincent Oliver
Editor David Burnie
Art Editor Peter Luff
Managing Editor Alan Buckingham

Traduzione di Paolo Ferrara e Marco Mezzalama
Edizione italiana a cura di Marco Maiocchi
Step-by-step programming for the Commodore 64 Book one

Via Matteo Bandello, 8
20123 Milano

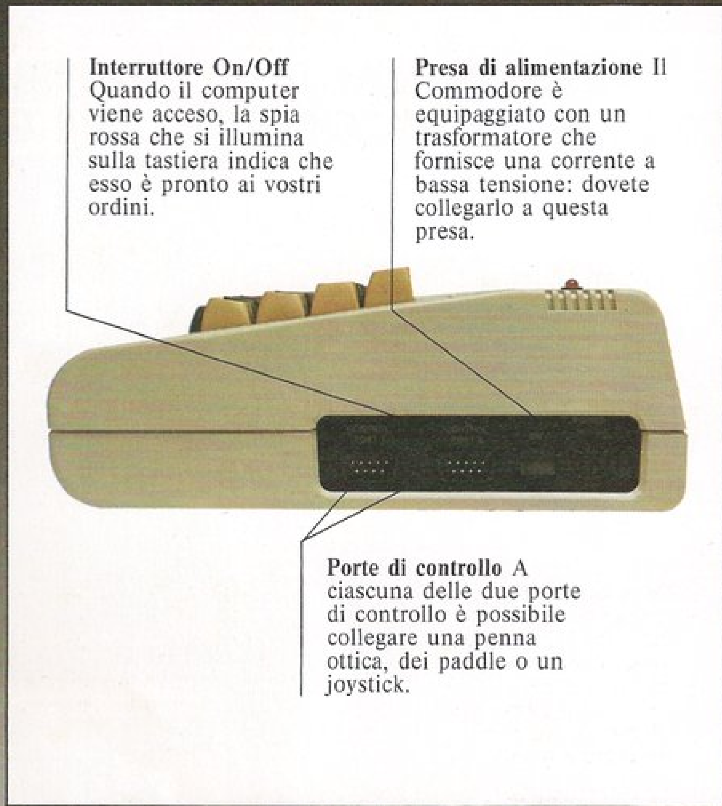
ISBN 88-377-0000-8
62a-I-85

Finito di stampare nel marzo 1985 da Officine Grafiche A. Mondadori Verona
Printed in Italy

IL COMMODORE 64

Il Commodore 64 è un personal computer estremamente versatile che fornisce comandi molto potenti, tra i quali la sintesi del suono su tre canali, grafici in alta e bassa risoluzione in 16 colori e animazioni in alta risoluzione utilizzando piccole figure mobili chiamate "sprite" (folletti).
Una volta che vi sarete impadroniti del semplice dialetto Commodore del linguaggio di programmazione BASIC, scoprirete che il vostro computer potrà regalarvi molte ore di interesse e divertimento.

Prese e connettori
Se vi ponete di fronte alla tastiera del Commodore, le prese che permettono di collegarlo ad altri dispositivi sono collocate sul lato destro e lungo il pannello posteriore del computer.
Girate il computer e date uno sguardo al retro: da sinistra a destra, il Commodore presenta un certo numero di prese di input e di output. Innanzi tutto il connettore (a 44 poli) per le cartucce di espansione (cartridges) in cui possono essere inserite delle memorie a sola lettura (ROM) preprogrammate; queste cartucce sono disponibili per diversi usi, tra cui giochi, programmi di utilità e altri linguaggi di programmazione. Le tre prese successive riguardano la trasmissione sullo schermo di segnali relativi all'immagine video. La prima di queste permette di variare il canale televisivo usato dal Commodore; la seconda è il connettore UHF che fornisce un segnale che può essere immesso direttamente nella presa d'antenna del vostro televisore. Se avete un monitor video Commodore, potete utilizzare invece la terza presa, la quale fornisce un segnale audio e video a colori. (Tutte le fotografie in questo libro sono state realizzate usando immagini prodotte su un monitor Commodore).
Dopo le tre prese c'è la porta seriale attraverso cui sono connesse la stampante e l'unità disco Commodore. Questa è seguita dall'interfaccia per cassette che è usata per collegare lo speciale registratore digitale a cassette Commodore C2N.
Infine, sulla destra, c'è una porta parallela. Questa è usata principalmente per interfacciare circuiti come un modem (MOdulatore-DEModulatore da usare per trasmissioni telefoniche) o una cartuccia per comunicazioni RS232.
La lista di connettori non finisce qui. Se girate il computer e guardate il lato destro potrete vedere un interruttore e ancora qualche presa di collegamento. Dalla parte anteriore a quella posteriore, i primi due connettori sono porte di controllo per i giochi; queste accettano input non digitali (analogici) da dispositivi come penne luminose, paddle e joystick. I segnali relativi sono poi convertiti in forma digitale per essere usati dai programmi. Dopo le porte di controllo c'è l'interruttore (on/off) principale di alimentazione e infine la presa di alimentazione stessa.

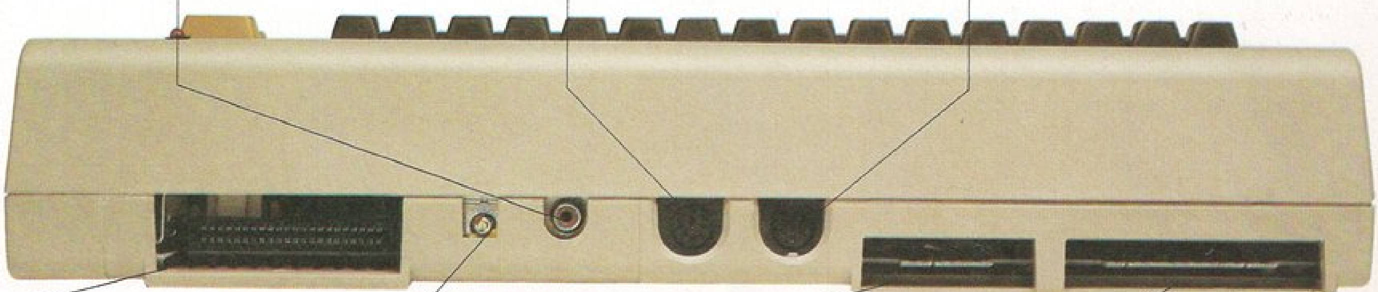




Presa UHF Serve per trasportare i segnali audio e video del Commodore nella presa d'antenna di un televisore.

Presa Audio/Video Attraverso questa presa è possibile connettere un monitor video, il quale fornisce immagini e suoni di alta qualità.

Porta di interfaccia Seriale Porta di interfaccia seriale per la stampante seriale standard e l'unità a disco Commodore 1541.



Presa per Cartucce Il Commodore dispone di molti altri linguaggi di programmazione e comandi memorizzati nelle ROM (Read Only Memory), che vanno collegate a questa presa.

Selettore di Canale Esso permette di variare leggermente il canale televisivo usato dal Commodore. Di solito non occorre nessuna regolazione.

Interfaccia per Cassette Essa connette il Commodore allo speciale registratore a cassetta C2N per memorizzare programmi e "file" di dati.

Porta Parallela Il Commodore può gestire molte periferiche per mezzo della porta parallela, tra le quali una stampante parallela Centronics.

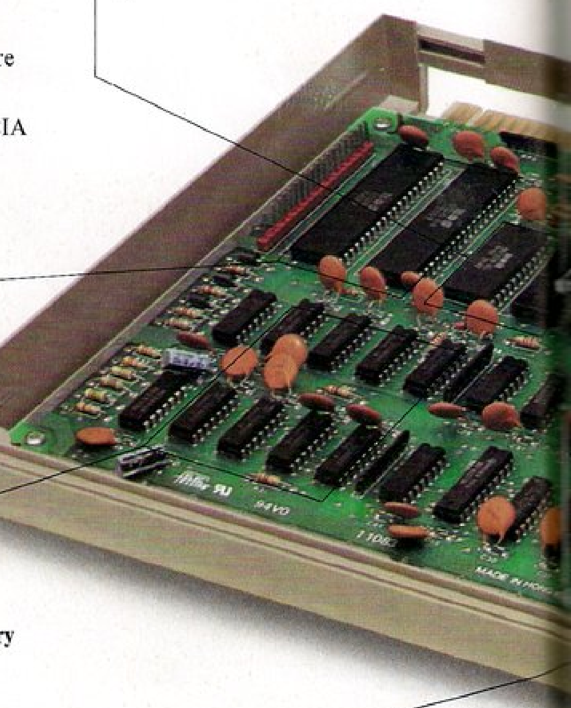


DENTRO IL COMPUTER

Il Commodore 64 è costituito da due piastre di circuiti. La prima sostiene la tastiera (guardate alle pagg. 10 e 11), mentre la seconda è la piastra principale, che copre interamente il fondo del contenitore. La parte superiore di quest'ultimo è progettata per essere rimovibile e permette alla piastra principale di essere esaminata in dettaglio.
La piastra contiene gli stessi elementi fondamentali che ogni personal computer usa: un microprocessore, vari tipi di memoria e alcuni chip di input/output.
Il microprocessore (CPU) permette i calcoli del computer e controlla le attività del resto della macchina: il Commodore 64 usa una CPU di tipo 6510.
La memoria del computer è divisa in due tipi principali: memoria ad accesso casuale o RAM (Random Access Memory) e memoria a sola lettura o ROM (Read Only Memory).
La RAM è anche chiamata "memoria volatile" poiché i suoi contenuti vanno persi quando viene interrotta l'alimentazione, si tratta cioè di una memoria di deposito temporaneo: le informazioni sono mantenute solo quando essa è alimentata.
La ROM, invece, è una memoria permanente: le istruzioni che essa contiene non sono cancellate quando il computer viene spento.

BASIC e codice macchina
I codici con cui lavorano i computer sono composti interamente di impulsi elettrici; il sistema di numerazione che è usato dai circuiti del computer è basato unicamente su due numeri, "0" e "1", dove "0" è rappresentato da "off" (nessun impulso) e "1" da "on" (presenza di un impulso): questo sistema è chiamato codice binario. In comune con molti personal computer, il Commodore tratta gruppi di otto cifre binarie (bit) alla volta; ogni gruppo di otto bit (un bit è un singolo "0" o "1") è chiamato "byte".
Il dato, in ogni singolo byte di memoria, rappresenta un carattere o simbolo della tastiera. Il Commodore 64 contiene 64K di memoria RAM e 16K di memoria ROM. Un "K" significa un Kilobyte, cioè un insieme di 1024 byte.
Il Commodore è programmabile molto facilmente con il linguaggio "ad alto livello" BASIC. Il BASIC sul Commodore è compatto e occupa solo 8K di ROM; questo significa che qualcuno dei comandi della macchina è accessibile solo in modo particolare, senza usare specifiche istruzioni BASIC. Prima che il computer possa agire sulle istruzioni fornitegli in BASIC, esso deve tradurle in codice macchina (il linguaggio che la CPU comprende); il BASIC è perciò più lento che il codice macchina, ma ha il vantaggio di essere molto più semplice da capire. Quando il computer è acceso, automaticamente seleziona il programma contenuto nella ROM del BASIC. Questo programma è chiamato "interprete" BASIC.

La catena di comandi di microchip Tutti i chip all'interno del Commodore formano una catena elettronica di comandi in cui la CPU si occupa di tutte le funzioni da eseguire. Il resto dei chip, comprese le RAM e le ROM, agiscono da supporto temporaneo o permanente delle informazioni necessarie alla CPU per gestire tutte le operazioni del computer.



ROM Kernel e ROM BASIC La ROM BASIC contiene le istruzioni necessarie per trasformare un programma in una forma interpretabile dalla CPU, il chip più importante. La ROM Kernel contiene le istruzioni per la comunicazione con le periferiche.

Il chip Tastiera/Porta Parallela Il Commodore utilizza due chip CIA 6526 di controllo per interfaccia parallela (CIA = Complex Interface Adapter) per interfacciarsi con la tastiera e la porta parallela appunto.

Random Access Memory (RAM) Otto chip di RAM forniscono 64K di memoria per contenere tutte le informazioni e i programmi che vengono memorizzati nel computer dopo che esso è stato acceso.

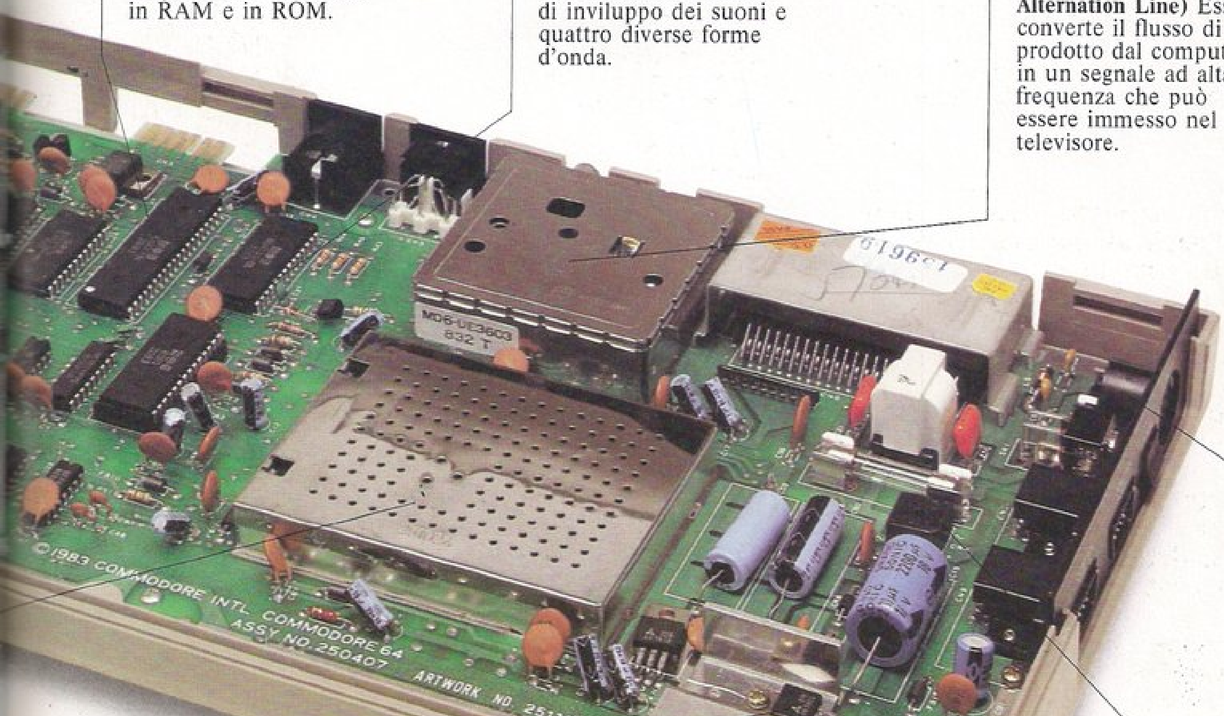
Video Interface Circuit (VIC) Il chip VIC gestisce la grafica in bassa e alta risoluzione, i colori e gli sprite.



Central Processing Unit (CPU) È la parte del computer che esegue le istruzioni. Il microprocessore effettua tutti i calcoli e gestisce l'attività del resto del computer utilizzando i programmi e le informazioni contenute in RAM e in ROM.

Sound Interface Device (SID) Il chip SID costituisce un completo sintetizzatore sonoro a tre canali con controllo di inviluppo dei suoni e quattro diverse forme d'onda.

Trasmettitore PAL (Phase Alternation Line) Esso converte il flusso di dati prodotto dal computer in un segnale ad alta frequenza che può essere immesso nel televisore.



Presa d'Alimentazione Questa è la presa per l'alimentazione in bassa tensione.

Regolatore di Tensione Esso evita fluttuazioni di tensione di alimentazione che potrebbero disturbare il funzionamento del computer.

Alimentatore Questa parte del computer converte la corrente alternata (AC) in bassa tensione, proveniente dal trasformatore, in corrente continua (DC) usata dai circuiti del computer stesso.

LA TASTIERA DEL COMMODORE

Il Commodore ha una tastiera di alta qualità che è ugualmente utilizzabile sia da chi programma con un dito solo che da chi è molto più veloce. Essa appare molto simile alla tastiera di una macchina per scrivere, ma il blocco centrale di lettere normali e tasti numerici è circondato da un certo numero di tasti extra che non si trovano in una macchina per scrivere.
I tasti possono essere suddivisi per funzione in tre gruppi: il blocco centrale con numeri e lettere, gli altri tasti scuri intorno e i quattro tasti chiari sulla destra del blocco principale.

I tasti carattere
Quando uno dei tasti del blocco centrale è premuto, produce un carattere sullo schermo.
Potete usare questi tasti per scrivere parole che il computer può riconoscere, cioè comandi o informazioni che volete che il computer usi mentre esegue un programma.
Potete notare che così come hanno una lettera o un numero stampati sopra, questi tasti hanno anche simboli o parole stampati frontalmente: i simboli costituiscono un insieme di caratteri grafici che possono essere fatti apparire sullo schermo invece delle lettere. I tasti numerici controllano i colori e possono essere usati per cambiare il colore delle parole e dei disegni: più avanti, in questo volume, è fornita la completa descrizione del modo in cui usare questi caratteri grafici e i controlli dei colori.

Tasti di cursore e di immissione
Sul lato destro del blocco principale di tasti, ci sono quattro tasti speciali che controllano il movimento del cursore sullo schermo e permettono le immissioni. Il Commodore permette di scrivere e correggere programmi (editing) in modo molto flessibile.
Le righe di un programma, per essere visualizzate o modificate, devono essere prima listate sullo schermo, e quindi raggiunte usando i due tasti chiamati CRSR che si trovano nell'angolo in basso della tastiera. Il tasto INST/DEL (INSerT/DELete), nell'angolo in alto a destra della tastiera, può essere usato per inserire e cancellare caratteri mentre il tasto CLR/HOME cancella lo schermo e sposta il cursore nell'angolo superiore sinistro.

Tasti funzione
Se premete uno dei quattro tasti marrone chiaro sulla destra del blocco principale, non apparirà nessun carattere sullo schermo perché i tasti funzione del Commodore sono progettati per essere usati esclusivamente con un programma.

RUN/STOP Questo tasto interrompe il programma in esecuzione e mostra il punto in cui è avvenuta l'interruzione.

CTRL Questo tasto ha una funzione simile a quella del tasto Commodore: permette di selezionare gli otto colori riportati sui tasti numerici da 1 a 8. Può anche essere usato con i tasti 9 e 0 per produrre caratteri in reverse.



Tasto Commodore La funzione principale di questo tasto è quella di selezionare (con i tasti numerici da 1 a 8) il secondo insieme di colori disponibili sul Commodore. Può essere usato con i tasti alfabetici per produrre dei caratteri speciali.

Tasti Numerici Oltre a produrre i numeri da 0 a 9, questi tasti possono anche essere premuti contemporaneamente al tasto CTRL o al tasto Commodore per selezionare uno dei 16 possibili colori.

Return Il Commodore non eseguirà la maggior parte dei comandi fino a quando non sarà premuto questo tasto. Può essere considerato equivalente al tasto di ritorno carrello di una macchina per scrivere.

Restore Se usato con il tasto RUN/STOP, inizializza il computer e cancella lo schermo.

CLR/HOME Questo è uno dei quattro tasti di controllo del cursore del Commodore. Premendo solo questo tasto il cursore si posiziona nell'angolo superiore sinistro dello schermo. Premendo CLR/HOME contemporaneamente al tasto SHIFT si può cancellare lo schermo da tastiera.

INST/DEL Questo tasto è usato nell'immissione di un programma. Premendo solo INST/DEL il cursore arretra e cancella un carattere sullo schermo; premendo anche il tasto SHIFT, al contrario, si inserisce in una riga lo spazio per un carattere.



Barra Spaziatrice Funziona nello stesso identico modo della barra spaziatrice di una macchina per scrivere.

Tasti Funzione Questi quattro tasti possono essere usati, anche con il tasto SHIFT, per avere a disposizione otto tasti particolari i quali, se premuti durante l'esecuzione di un programma, possono essere utilizzati per controllare determinate operazioni.

SHIFT e SHIFT LOCK Questi tasti permettono di utilizzare i simboli speciali riportati sui tasti alfabetici e, se usati assieme al tasto Commodore, commutano l'insieme dei caratteri che si possono visualizzare.

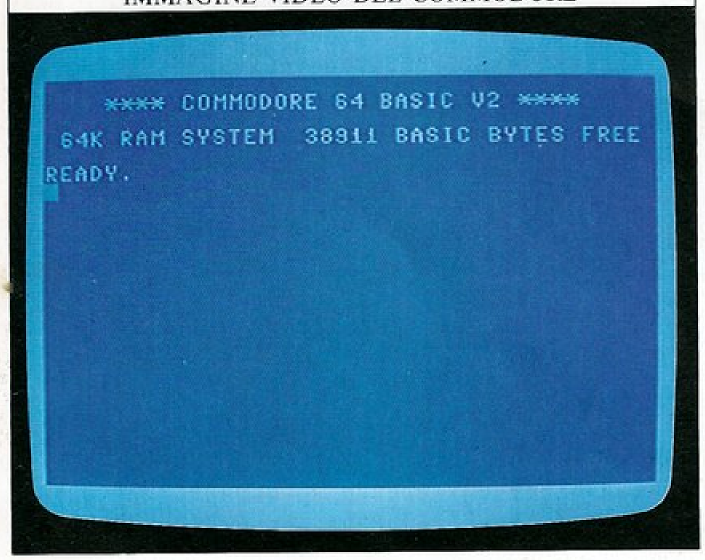
Tasti del Controllo del Cursore Questi due tasti, usati con o senza il tasto SHIFT, permettono di muovere il cursore in una direzione qualsiasi sullo schermo. Sono molto utili quando dovete immettere o correggere un programma.

INSTALLAZIONE

L'installazione del Commodore richiede due fasi distinte. Per prima cosa, dovete collegare il computer alle sue periferiche, quindi è necessario regolare il vostro monitor (o televisore) per ottenere il miglior risultato sullo schermo. La prima parte è semplice, la seconda invece può richiedere più tempo.
Il sistema, così come è fornito, comprende l'unità principale (la tastiera, la quale contiene tutti i circuiti del computer), un trasformatore di alimentazione e un cavo d'antenna per collegare il computer ad un televisore.
Il trasformatore di alimentazione ha due cavi: la spina DIN deve essere collegata alla presa di alimentazione del computer, che si trova sul lato destro del contenitore, mentre l'altro cavo è quello di rete, che deve essere collegato ad una presa della rete elettrica.
Se utilizzate un televisore, scollegate il cavo dell'antenna e sostituitelo con il cavo televisivo del Commodore; l'altro capo di quest'ultimo deve essere inserito nella presa UHF sul pannello posteriore del computer. Se usate un monitor Commodore, invece, collegatelo alla presa Audio/Video.
Quando accendete il computer tramite l'interruttore di alimentazione, deve accendersi la spia in alto a destra, sulla tastiera. Se state usando un televisore, accendetelo e selezionate un canale libero (sarebbe meglio poter selezionare un canale permanentemente destinato al computer).

Sintonizzazione
Quando il computer è acceso produce un segnale video che dovreste riuscire a ricevere sul vostro televisore. Dopo qualche esperimento con il controllo di sintonia dovreste vedere questa immagine, o una simile, sul vostro schermo:

IMMAGINE VIDEO DEL COMMODORE



Se non riuscite a vedere questa immagine, probabilmente non avete connesso il computer in modo corretto: controllate ancora tutti i cavi e le connessioni. Se usate un televisore a colori, dovete ottenere dei colori identici a quelli mostrati; se, invece, i colori risultassero essere molto diversi, provate a sintonizzare meglio l'apparecchio: quando un televisore a colori è leggermente fuori sintonia, può produrre un'immagine nitida in bianco e nero, senza però nessuna traccia di colore. Se avete un monitor, esso sarà già sintonizzato e, modificando alcuni controlli, potrete selezionare la migliore visualizzazione possibile dei colori.
Qualche volta, con un televisore, l'immagine non risulterà perfettamente centrata, ma voi potete correggere questa imperfezione utilizzando i normali controlli di regolazione orizzontale e verticale. Ricordatevi anche che il Commodore non possiede un altoparlante incorporato, ma usa quello del televisore: quando scrivete dei programmi che producono suoni, controllate che il volume del vostro apparecchio sia regolato in modo da poter ascoltare i suoni prodotti dal computer.
Dopo aver installato il vostro Commodore, noterete che l'immagine iniziale contiene la riga:

38911 BASIC BYTES FREE

Questo numero vi dice quanta memoria il computer rende disponibile per memorizzare i programmi BASIC e i dati che questi programmi utilizzano. Un byte di memoria può immagazzinare un carattere, una lettera o un simbolo. Non appena iniziate a programmare, la memoria comincerà ad essere usata (benché l'immagine sullo schermo non lo mostrerà). Poiché i vostri programmi sono conservati in memoria solo fino a che il computer è alimentato, avrete a disposizione lo stesso numero di bytes di memoria ogni volta che lo accenderete.

La combinazione dei colori nel Commodore
Oltre ai colori blu e azzurro visibili nell'immagine iniziale, il Commodore è in grado di produrre molti altri colori. Ce ne sono 16 in tutto, che danno luogo a un gran numero di combinazioni che possono essere visualizzate sullo schermo. Come scoprirete ben presto, alcune di queste, in particolare le combinazioni di colori forti, appariranno molto gradevoli; quelle in cui compaiono solo colori chiari, invece, spesso ottengono meno successo. Alcune combinazioni di colori possono inoltre produrre "frange" o ombreggiature sullo schermo che rendono il testo difficilmente leggibile: questo fatto non è dovuto ad una sintonizzazione difettosa, e il miglior modo per aggirare il problema è cambiare colori. Le combinazioni usate in questo libro vi daranno un'idea di quali colori possono essere usati con un risultato gradevole.

Come connettere le periferiche
Le periferiche sono degli accessori, registratori a cassetta, unità a disco e stampanti, che potete collegare al computer. Oltre ai monitor video, quella che probabilmente userete più di frequente è il registratore a cassetta Commodore o l'unità a disco: entrambe vengono usate per memorizzare e richiamare i programmi; la

Commodore produce registratori a cassetta e unità a disco appositamente progettate per essere usate con il vostro Computer. Una unità a disco svolge la stessa funzione di un registratore a cassetta ma, oltre a salvare e caricare programmi più velocemente, permette l'accesso immediato ad ogni programma su disco senza dover effettuare una ricerca dello stesso dall'inizio del nastro, come occorre fare con un registratore a cassetta. Questa riduzione del tempo di accesso è molto utile se volete registrare e riutilizzare frequentemente i vostri programmi; a pag. 58 viene spiegato in dettaglio come usare un registratore a cassetta o una unità a disco. In generale, quando state facendo un collegamento, l'inserimento di un connettore in una presa non deve offrire alcuna resistenza. Se ciò invece accade, state probabilmente inserendo il connettore in una presa sbagliata; se non è così, controllate che non ci siano sbavature sulla presa e che nessun piedino del connettore sia piegato. Quando dovete staccare un connettore, non fatelo mai tirando il cavo: potreste danneggiarlo oppure interrompere i collegamenti all'interno del connettore, o rompere il connettore stesso.



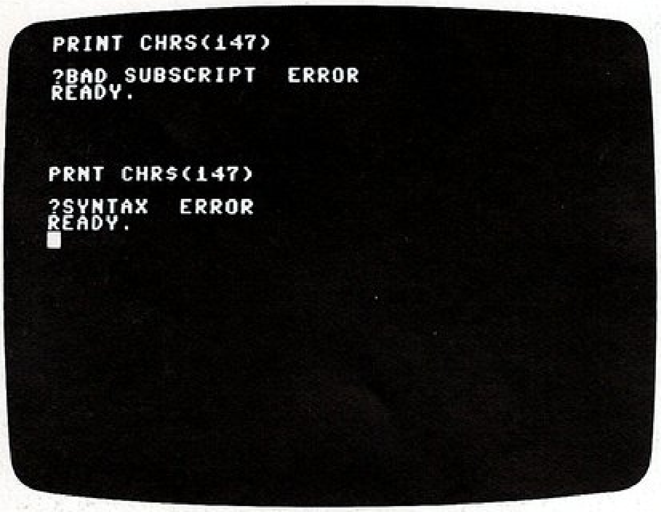
Immagini e listati a colori I due colori (AZZURRO E BLU) che appaiono sullo schermo del Commodore possono essere variati sia direttamente, attraverso la tastiera, sia indirettamente, per mezzo di un programma. I listati dei programmi, normalmente, saranno visualizzati nelle due tonalità di blu. Per rendere più leggibile il testo dei programmi presentati in questo libro, il computer è stato programmato per visualizzare i listati in bianco su nero (il metodo per fare ciò è illustrato a pag. 32). Questo fatto non crea nessun problema: voi potete visualizzarli in un colore qualsiasi.

COME COMINCIARE

Dopo aver installato il vostro Commodore, potreste avere la tentazione di premere un tasto. Fatelo pure, non potreste comunque causare nessun danno. Nella maggioranza dei casi, schiacciando un tasto, fate apparire sullo schermo il simbolo corrispondente e, essendo riusciti a far visualizzare qualcosa al computer, potreste anche voler sapere come cancellarlo. Il metodo più semplice è quello di mantenere premuto SHIFT e premere CLR (il secondo tasto dalla destra nella riga superiore). Questa operazione cancella tutto lo schermo. Per reinizializzare il computer, in modo che ogni comando che potete aver dato venga annullato, spegnetelo e poi riaccendetelo. Dovete però stare attenti, se decidete di compiere questa operazione, perché essa cancella qualsiasi programma caricato in memoria.
Un altro sistema per cancellare lo schermo è quello di immettere il comando PRINT CHR$(147) e quindi di premere il tasto RETURN.

Come dare esattamente le giuste istruzioni
È importante ricordare che il computer può obbedire solo alle istruzioni che sono perfettamente corrette. Se immettete PRINT CHR$(147) lo schermo verrà cancellato, ma se immettete, per esempio, PRINT CHRS(147), potrete solo ottenere un messaggio di errore, uno di quelli che il computer mantiene memorizzati nella sua memoria permanente. Ciò accade perché il computer non riesce a comprendere quello che avete immesso. Ecco la tipica reazione del computer alle istruzioni non corrette:

MESSAGGIO DI ERRORE



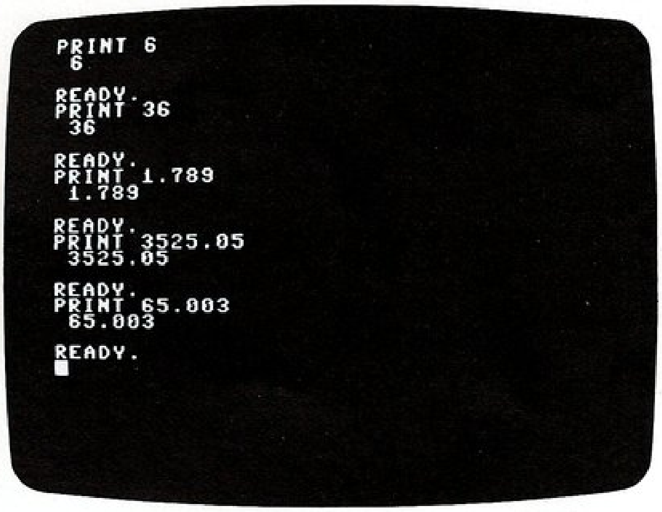
Così, se nelle pagine successive il computer si rifiuterà di obbedire alle vostre istruzioni, potete essere certi che questo è dovuto al fatto che avete immesso un comando che esso non riesce ad interpretare.

Come visualizzare qualcosa sullo schermo
Ora cancellate lo schermo e immettete questa riga:

```
PRINT 6
```

Se premete il tasto RETURN, il numero 6 apparirà sulla riga successiva dello schermo. Il computer ha risposto al vostro comando; PRINT (STAMPA) non ha niente a che fare con carta e inchiostro: è solo un modo per dire al computer di visualizzare qualcosa sullo schermo. Provate a dare lo stesso comando, nello stesso modo ma con altri numeri.
Non ha importanza se, per chiarezza, lasciate uno spazio tra PRINT e il numero, o meno: il computer può ancora capire l'istruzione.

VISUALIZZARE DEI NUMERI

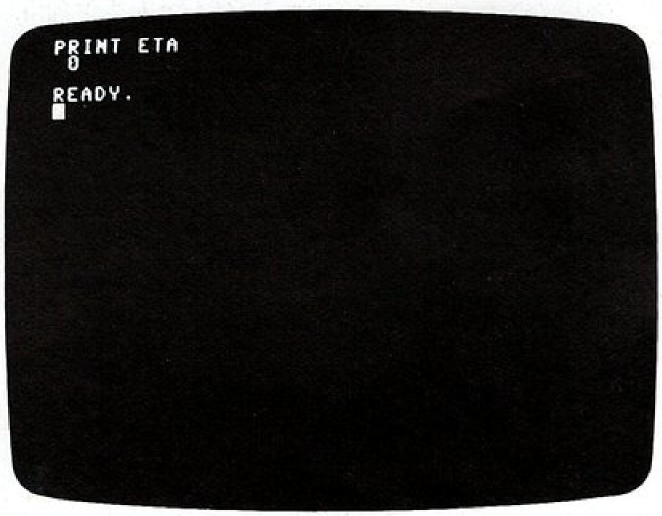


Ora cancellate lo schermo premendo SHIFT e CLR e quindi immettete:

```
PRINT ETA
```

Il computer risponde visualizzando uno zero, e non la parola ETA, come avreste potuto aspettarvi:

VISUALIZZARE DELLE PAROLE



Il computer ha cercato nella sua memoria una "variabile chiamata ETA e, poiché non è riuscito a trovarla (in

quanto non gli era stato detto nulla in merito), esso la crea e le assegna il valore zero. Questo valore è quello che vedete visualizzato.

Che cosa è una variabile?

Senza cancellare lo schermo immettete ora l'istruzione in modo differente:

PRINT "ETA"

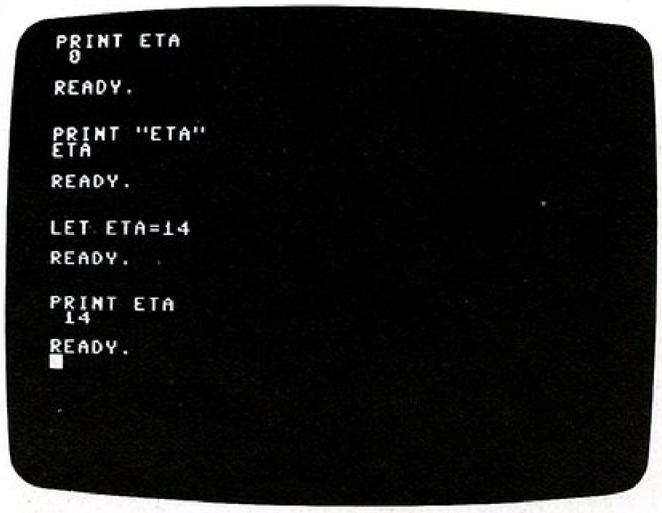
Questa volta il computer visualizza proprio la parola ETA sulla successiva linea dello schermo. Avete così scoperto che, per il computer, ETA (scritta in questo modo) e "ETA" (tra virgolette) hanno significati completamente diversi. Esso tratta ogni lettera o ogni gruppo di lettere come nomi di variabili: una variabile è semplicemente un nome che identifica un numero immagazzinato in memoria.

Ora siete in grado di capire perché l'istruzione PRINT ETA ha fornito un risultato apparentemente strano: il computer interpreta ETA non come una parola, ma come il nome di un particolare dato in memoria; esso cerca nella sua memoria un numero chiamato ETA ma, poiché non avete ancora memorizzato nulla, non lo può trovare.

Variabili numeriche e variabili stringa

Per rendere più comprensibile al computer il comando PRINT ETA, assegnate ad ETA un valore. Provate ad immettere le seguenti righe (premendo il tasto RETURN dopo ogni riga):

COME USARE LET



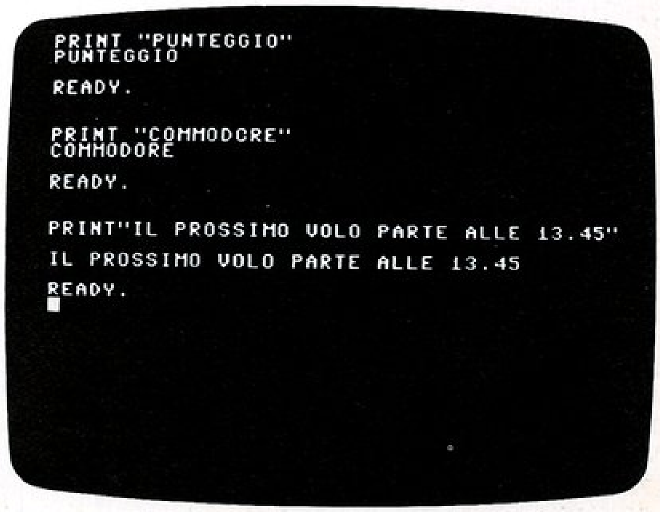
Ora "ETA" identifica un dato, il numero 14.

LET è un comando che assegna un nome ed un valore ad una locazione di memoria; ogni volta che chiederete al computer di visualizzare ETA, esso vi mostrerà l'ultimo dato che vi avete memorizzato.

"ETA" è una stringa mentre ETA, che rappresenta sempre un numero, è detta variabile numerica.

Il computer visualizza qualsiasi cosa poniate tra virgolette esattamente come voi la scrivete. Provate. Potete usare qualsiasi carattere della tastiera, lettere, numeri, simboli matematici e segni di punteggiatura, senza superare la lunghezza massima per un comando PRINT (due righe):

VISUALIZZARE DELLE STRINGHE



I caratteri tra virgolette costituiscono una stringa. Così come una variabile numerica è memorizzata nel computer e identificata con un nome per mezzo di una variabile numerica, una stringa è memorizzata e identificata grazie ad una variabile stringa, la quale deve sempre terminare con il simbolo "$". A$, NUMBER$, PRICE$ e CITY$ sono tutte variabili stringa. Nella riga:

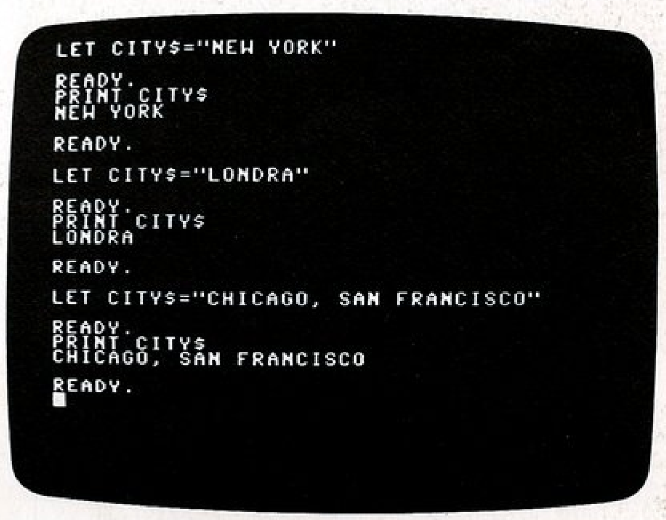
LET CITY$ = "NEW YORK"

per esempio, CITY$ è una variabile stringa e NEW YORK è la stringa che essa identifica. Cancellate di nuovo lo schermo e immettete la riga precedente da tastiera; se quindi scrivete:

PRINT CITY$

il computer mostrerà il contenuto della variabile stringa CITY$. Come per una variabile numerica, il comando LET vi permette di memorizzare una stringa nella memoria del computer. Anche in questo caso il computer manterrà in memoria solo l'ultima versione della stringa indicata da una particolare variabile, perciò potete modificare CITY$ come volete:

MODIFICARE UNA STRINGA



UTILIZZARE I COLORI

Una delle caratteristiche fondamentali del Commodore 64 è la sua capacità di produrre colori molto vivaci. La macchina può visualizzare 16 colori che possono essere selezionati direttamente dalla tastiera, senza bisogno di utilizzare un particolare programma. In queste due pagine potrete imparare come cambiare il colore del testo sullo schermo e, come potrete vedere, ci sono un certo numero di modi differenti per farlo. Quando accenderete il computer l'immagine è blu e azzurra o ciano ma, molte delle illustrazioni in questo libro mostrano un testo bianco su uno sfondo nero, che è più facile da leggere dell'immagine normale. Imparerete a produrre testi in bianco in un attimo, ma se volete anche cambiare lo sfondo e farlo diventare nero, immettete:

```
POKE 53280,0:POKE 53281,0
```

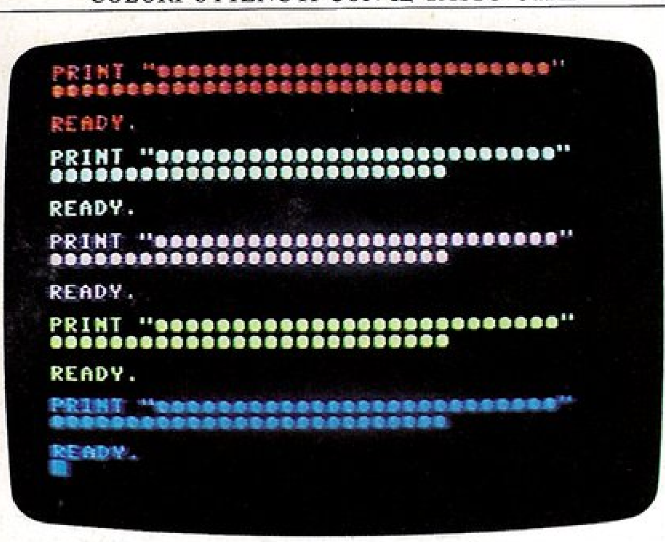
e quindi premete RETURN. Potrete vedere come funzionano questi comandi BASIC alle pagg. 32 e 33.

Come cambiare i colori da tastiera

Se osservate la tastiera del Commodore, potete notare che sulla parte anteriore dei tasti numerici da 1 a 8 sono stampati i nomi di otto colori.

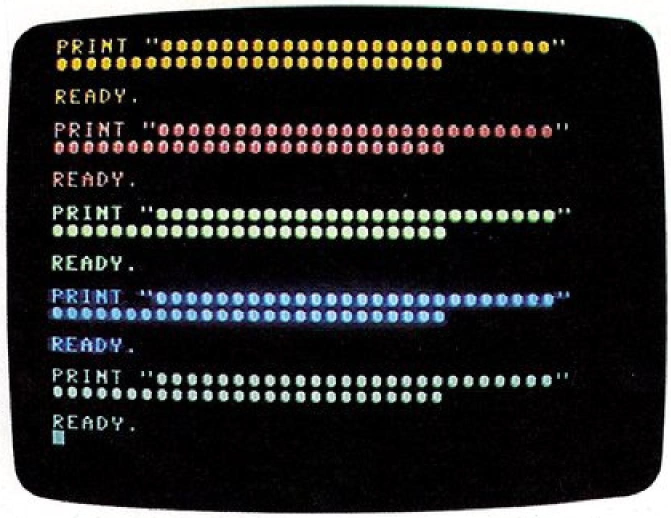
Questi tasti sono usati assieme al tasto CTRL (primo tasto a sinistra della seconda riga a partire dall'alto) per selezionare 8 dei 16 colori disponibili. Provate con questo semplice esercizio: mantenete premuto il tasto CTRL, e quindi schiacciate il tasto indicato con BLK (tasto 1); noterete che ora il cursore lampeggiante sullo schermo è diventato nero e che tutte le volte che digitate qualcosa, questo appare in nero. Ugualmente, se viene premuto qualsiasi altro tasto numerico insieme al tasto CTRL, il colore del cursore cambierà in quello indicato dal tasto numerico relativo. Questa schermata mostra una serie di colori selezionati con il tasto CTRL (il circoletto pieno è selezionato schiacciando SHIFT e Q):

COLORI OTTENUTI CON IL TASTO CTRL



Gli altri colori disponibili sul Commodore possono essere selezionati nello stesso modo, ma premendo il tasto indicato C= (tasto Commodore) invece di CTRL. La prossima schermata mostra alcuni di questi colori, e la tabella seguente indica come potete selezionarli:

TASTI COLORE DEL COMMODORE



I COLORI DEL COMMODORE

I colori prodotti dai tasti numerici sono selezionati mediante l'uso dei tasti CTRL o Commodore (C=).

Numero tasto	Colori prodotti con CTRL	Colori prodotti con C=
1	Nero	Arancione
2	Bianco	Marrone
3	Rosso	Rosa
4	Ciano	Grigio scuro
5	Porpora	Grigio
6	Verde	Verde chiaro
7	Blu	Azzurro
8	Giallo	Grigio chiaro

Come cambiare i colori con i simboli di controllo

Così come usate direttamente i colori del Commodore, potete anche immettere simboli speciali di controllo in una stringa col comando LET, in modo che i colori vengano modificati solo quando il computer visualizza una

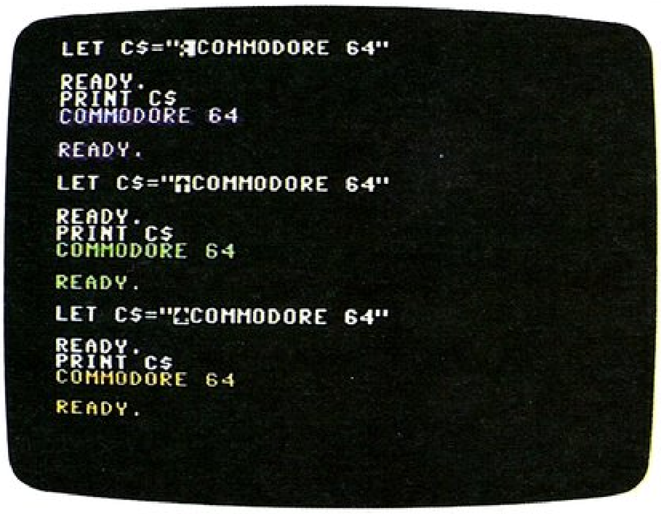
SIMBOLI DI CONTROLLO DEI COLORI

Ognuno dei 16 colori del Commodore, può essere immesso come un carattere rappresentato da un simbolo di controllo colore.

Colore	Tasti di selezione	Simbolo di controllo	Colore	Tasti di selezione	Simbolo di controllo
Nero	CTRL 1		Arancione	C=1	
Bianco	CTRL 2		Marrone	C=2	
Rosso	CTRL 3		Rosa	C=3	
Ciano	CTRL 4		Grigio scuro	C=4	
Porpora	CTRL 5		Grigio	C=5	
Verde	CTRL 6		Verde chiaro	C=6	
Blu	CTRL 7		Azzurro	C=7	
Giallo	CTRL 8		Grigio chiaro	C=8	

stringa; caratteristica questa che può essere usata in un programma. Questo è cio che potrete vedere sullo schermo:

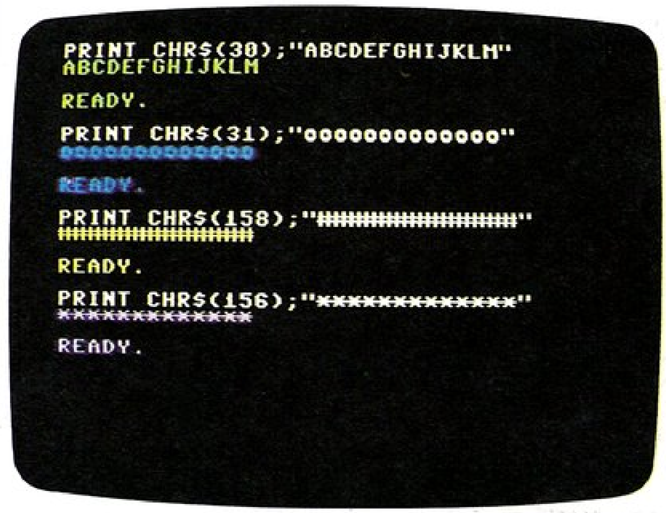
CAMBIARE COLORE CON I SIMBOLI DI CONTROLLO



I tasti usati per inserire gli speciali simboli di controllo sono gli stessi che avete già usato nel modo diretto: i tasti numerici più CTRL o C=; ma, come potete vedere, quello che appare nella stringa immessa con questo metodo non è in nessun modo collegato ai colori scelti: ogni simbolo di controllo è solo uno dei molti caratteri grafici che sono permanentemente memorizzati nel Commodore.
Benché sia facile utilizzare i tasti relativi ai simboli di controllo colore, copiarli dal listato di un programma può essere difficoltoso. Per eliminare ogni tipo di confusione sui comandi di cambiamento dei colori, per i programmi di questo libro verrà spesso usato un terzo modo per selezionare i colori, quello che usa codici numerici di controllo.

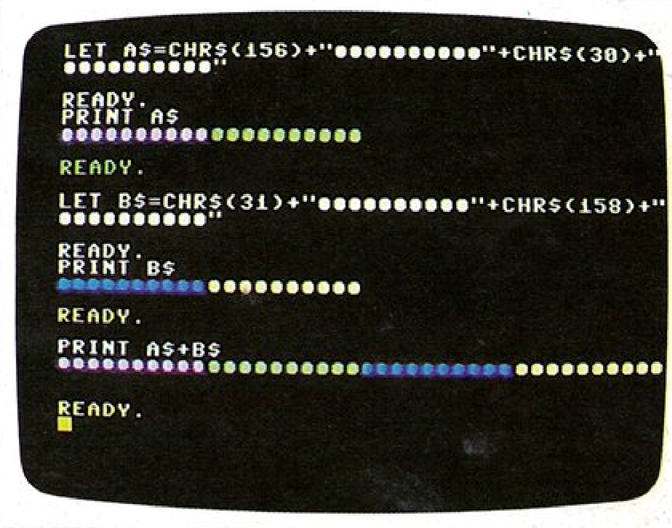
Come cambiare i colori con i codici numerici
Il Commodore usa uno speciale codice chiamato ASCII (American Standard Code for Information Interchange) per rappresentare l'insieme dei caratteri che possono essere visualizzati sullo schermo. Così, per esempio, la lettera maiuscola "A", è rappresentata nel codice ASCII dal numero 65, la lettera "B" dal numero 66 e così via; a pag. 61 potete trovare una lista completa di questi codici. Oltre che rappresentare tutti i caratteri visualizzabili, i codici ASCII attivano anche operazioni come il movimento del cursore, il cambiamento di linea e il controllo dei colori. Nel BASIC del Commodore esiste una speciale parola chiave che può essere usata con PRINT per produrre un carattere o una operazione, se si conosce il codice ASCII relativo: si tratta di CHR$. Avete già incontrato questa funzione nell'istruzione PRINT CHR$(147) che cancella lo schermo. Come potete vedere dalla tabella riportata in questa pagina, i codici ASCII di controllo colore non formano una serie continua, tuttavia, malgrado questo svantaggio, usando CHR$ ci sono meno possibilità di far nascere errori di interpretazione nei programmi che copierete, poiché i numeri sono più facilmente riconoscibili rispetto ai simboli grafici. Questo è il metodo di cambiamento dei colori del testo che sarà usato più spesso in questo libro; se voi volete usarne altri, comunque, il programma funzionerà ugualmente. Ecco alcuni cambiamenti di colore effettuati con CHR$:

CAMBIARE COLORE CON **CHR$**



Come usare CHR$ nelle stringhe
Come usate LET per inserire simboli di controllo colore in una stringa, così potete inserire i codici per CHR$. Anche qui essi sono attivi solo dopo l'esecuzione di PRINT:

CAMBIARE COLORE CON **CHR$** E **LET**



CODICI DI CONTROLLO COLORE ASCII

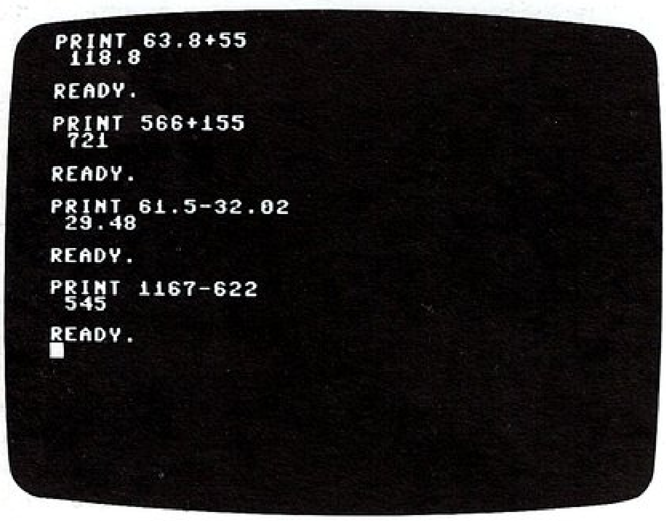
Il comando CHR$ può essere usato con un codice ASCII direttamente oppure in un programma per ottenere i colori che sono accessibili per mezzo del tasto CTRL o del tasto Commodore.

Colore	Codice ASCII	Colore	Codice ASCII
Nero	CHR$(144)	Arancione	CHR$(129)
Bianco	CHR$(5)	Marrone	CHR$(149)
Rosso	CHR$(28)	Rosa	CHR$(150)
Ciano	CHR$(159)	Grigio scuro	CHR$(151)
Porpora	CHR$(156)	Grigio	CHR$(152)
Verde	CHR$(30)	Verde chiaro	CHR$(153)
Blu	CHR$(31)	Azzurro	CHR$(154)
Giallo	CHR$(158)	Grigio chiaro	CHR$(155)
Reverse att.	CHR$(18)	Reverse dis.	CHR$(146)

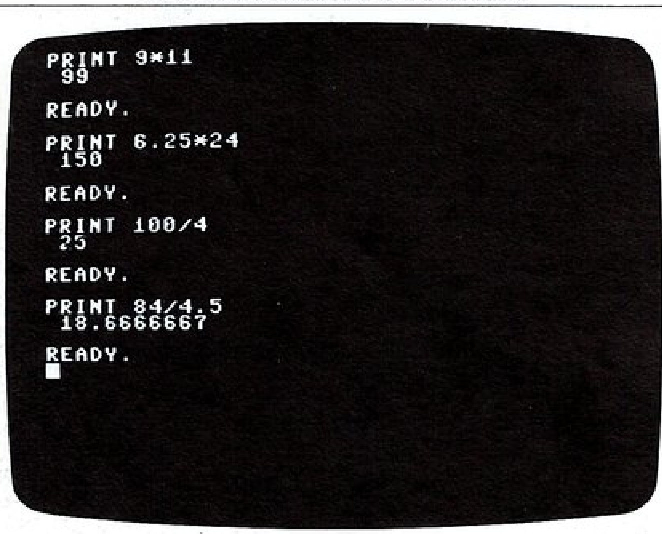
I CALCOLI DEL COMPUTER

Il comando PRINT non è limitato solo alla visualizzazione di caratteri sullo schermo.
Esso può anche essere usato insieme alle quattro operazioni aritmetiche, addizione, sottrazione, divisione e moltiplicazione, per eseguire dei calcoli che potete vedere sullo schermo.
Consideriamo innanzitutto l'addizione: il tasto più (+) è il quinto tasto da destra nella prima fila in alto; per sommare insieme due numeri, si usa semplicemente PRINT seguito dalla espressione del calcolo.
La sottrazione è effettuata allo stesso modo. Il segno meno, che è il trattino usato nel testo, si trova vicino al tasto "+".
La schermata seguente mostra alcune semplici operazioni di addizione e divisione:

ADDIZIONI E SOTTRAZIONI



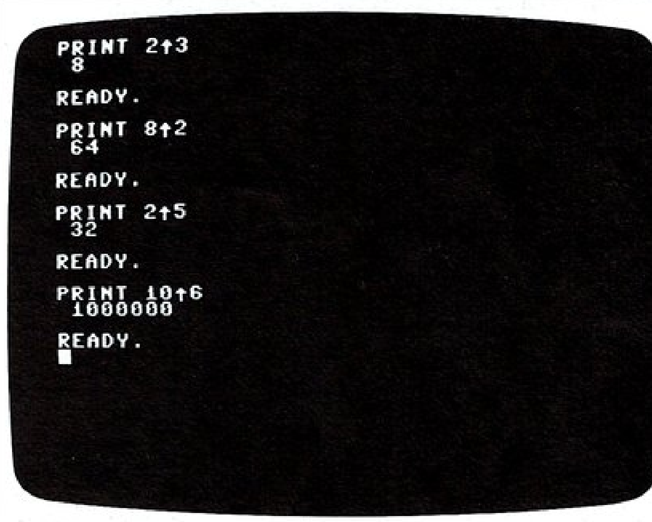
MOLTIPLICAZIONI E DIVISIONI



La moltiplicazione non è effettuata con il familiare segno "x", ma con l'asterisco (*); quest'ultimo si trova sotto il tasto del meno. La divisione usa la sbarra obliqua (/) che è vicino al tasto SHIFT di destra; in 24/8, per esempio, il numero a sinistra è diviso per il numero a destra. Come potete vedere nella seconda schermata, quando una divisione produce un risultato periodico, o comunque con molte cifre decimali, come in 84/4.5, il computer dà il risultato su 9 cifre.

Elevazione a potenza e radice quadrata
Oltre a queste familiari operazioni aritmetiche, potete moltiplicare un numero per sè stesso un certo numero di volte (operazione chiamata elevazione a potenza) e calcolarne la radice quadrata. Per esempio 2^3 è equivalente a 2 moltiplicato per sè stesso 3 volte, cioè a 8. La tastiera non produce esponenti (come 3 in 2^3), così dovete utilizzare il simbolo della freccia verso l'alto (↑) situato sotto il tasto CLR. 2^3 deve essere immesso nel computer come 2↑3. Ecco alcuni esempi:

ESPONENTI



Il computer vi permette anche di trovare la radice quadrata di un numero; questa volta non c'è un singolo tasto che realizza il calcolo, occorre invece immettere un comando come questo:

```
PRINT SQR(2)
```

Assicuratevi di usare le parentesi tonde che si trovano sui tasti numerici 8 e 9, e non le parentesi quadre alla sinistra del tasto RETURN. Poiché le parentesi sono sopra i simboli dei due numeri, dovete premere il tasto SHIFT insieme a quello del simbolo della parentesi per selezionarle durante l'impostazione della riga di comando.
Quando premete RETURN il computer visualizza il risultato del calcolo; se provate ad applicare questa operazione ad un numero negativo, cioè preceduto dal segno meno, il computer produrrà in risposta un messaggio di errore, per far capire l'impossibilità matematica di tale operazione.

Come specificare una sequenza di calcoli
Usando lo stesso comando PRINT, potete effettuare

una certa sequenza di calcoli. Provate prima con le sole operazioni di addizione e sottrazione. Scoprirete che la memoria della macchina sembra inesauribile:

UNA SEQUENZA DI CALCOLI

```
PRINT 3+11+2-1.9+15+0.5
 29.6

READY.

PRINT 48-42+16-2
 20

READY.

PRINT 11.007+0.089-1.2+37.5-1.2
 46.196

READY.

PRINT 1200+3570-2500+96010
 98280

READY.
█
```

Potete introdurre le cifre per ogni calcolo in qualsiasi ordine: il risultato sarà sempre lo stesso.
Tuttavia quando aggiungete moltiplicazioni e divisioni, potete ottenere risultati apparentemente strani. Guardate la prossima schermata e provate ad eseguire i calcoli a mano.
Supponiamo che vogliate addizionare due numeri e dividere il risultato per 2. L'ordine in cui i numeri sono addizionati non dovrebbe creare alcuna differenza sul risultato, ma succede che:

STESSI CALCOLI MA RISULTATI DIFFERENTI

```
PRINT 3+4/2
 5

READY.

PRINT 4+3/2
 5.5

READY.

PRINT (3+4)/2
 3.5

READY.

PRINT (4+3)/2
 3.5

READY.
█
```

Se scrivere 3+4 è esattamente lo stesso che scrivere 4+3, perché allora la successiva divisione per 2 dovrebbe produrre risultati differenti? La ragione è che il computer non necessariamente esegue i calcoli nell'ordine con cui sono visualizzati sullo schermo; esso per prima cosa, esegue l'elevazione a potenza, quindi moltiplicazione e divisione ed infine addizione e sottrazione. Così in PRINT 3+4/2 prima 4 è diviso per 2, poi viene sommato 3 al risultato. Nell'istruzione PRINT 4+3/2 invece, prima 3 è diviso per 2, poi 4 è sommato al risultato. Il problema richiedeva di sommare due numeri insieme e dividere il risultato per due. Né l'uno né l'altro di questi esempi fa al vostro caso. Potete però cambiare l'ordine in cui il computer realizza i calcoli usando delle coppie di parentesi tonde, come nei due esempi finali sullo schermo.
In questo caso, l'addizione è eseguita per prima e quindi il risultato è diviso per 2.

Quali sono i limiti del computer?
Ci sono dei limiti ai numeri che il computer può utilizzare e questi limiti sono di due tipi: di dimensione e di precisione. La limitazione nella dimensione non è probabilmente un inconveniente: i numeri con il punto decimale possono assumere qualsiasi valore nell'intervallo da $1x10^{38}$ (1 seguito da 38 zeri) a $-1x10^{38}$ (−1 seguito da 38 zeri), mentre gli interi possono assumere qualsiasi valore compreso tra 32767 e −32768.
Anche la precisione di questi due tipi di numeri è trattata in modo leggermente differente in ognuno dei due casi. Benché i numeri con il punto decimale possano arrivare sino ad 1 seguito da 38 zeri, il computer memorizza solo le prime nove cifre, mentre il resto è posto a zero; questa accuratezza di nove cifre è adeguata per la maggior parte delle applicazioni.
I numeri interi sono invece memorizzati con precisione completa, perciò all'interno del loro insieme di valori, il computer registra gli interi con una accuratezza migliore di 1 parte su 5000!
Incontrerete altre stranezze quando utilizzerete numeri molto grandi: il Commodore non li potrà visualizzare nel modo in cui voi li immettete da tastiera. Per esempio, PRINT 1000000000000 produce 1E+12 sullo schermo (la E sta ad indicare la presenza dell'esponente). Questo è solo un modo sintetico per visualizzare $1x10^{12}$, 1 seguito da 12 zeri, cioè il numero che avete immesso.
Provate ad inserire alcuni numeri molto grandi ed eseguite dei calcoli usando il comando PRINT; guardate come il computer visualizza i risultati:

VISUALIZZAZIONE DI NUMERI MOLTO GRANDI

```
PRINT 10000
 10000

READY.
PRINT 1000000
 1000000

READY.
PRINT 100000000
 100000000

READY.
PRINT 10000000000
 1E+10

READY.
PRINT 1000000000000
 1E+12

READY.
PRINT 2E20*2E20

?OVERFLOW  ERROR
READY.
█
```

COME SCRIVERE IL VOSTRO PRIMO PROGRAMMA

Fino ad ora avete dato al vostro Commodore comandi a cui esso può rispondere immediatamente. Questi comandi sono stati molto semplici: in molti casi avreste fatto prima senza l'uso del computer. In realtà dei semplici comandi non sono un programma per il computer; esso legge ogni comando, lo esegue e quindi lo dimentica. Un programma, invece, è costituito da una lista ordinata di istruzioni che il computer può immagazzinare nella sua memoria, e può eseguire quando volete che un lungo e complesso insieme di istruzioni possa essere eseguito mediante la semplice pressione di un tasto.

Da singoli comandi a righe di programma

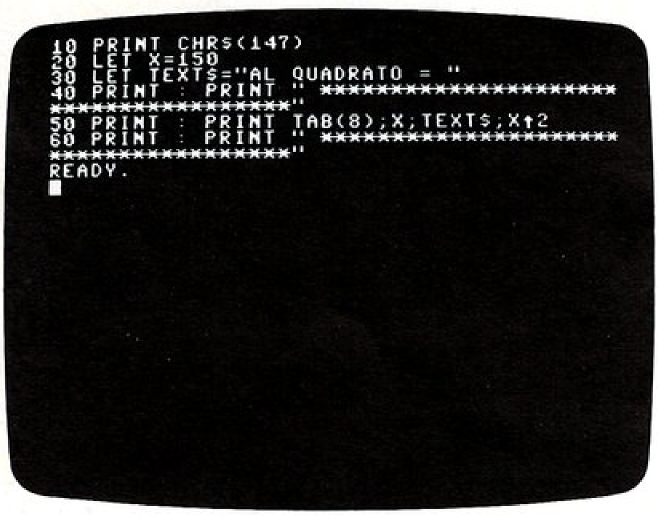
Deciso il compito che volete far eseguire al vostro Commodore, il passo successivo consiste nello scrivere il programma mediante istruzioni che il computer possa comprendere.

Il Commodore, come la maggior parte dei personal computer, usa un linguaggio di programmazione chiamato BASIC (Beginners' All-purpose Symbolic Instruction Code); il BASIC è un esempio di linguaggio ad alto livello, un linguaggio composto da parole e simboli con cui voi, l'utente, avete già familiarità. Si tratta, perciò, di uno dei linguaggi per la programmazione dei computer più facile da imparare, come potrete scoprire voi stessi.

L'essenza di un programma sta nel fatto che esso è memorizzato nella memoria del computer.

I comandi che avete inserito fino ad ora possono essere tradotti in programmi aggiungendo semplicemente dei numeri di riga. Ecco un semplice programma di sei righe:

PROGRAMMA "ELEVAZIONE A QUADRATO"

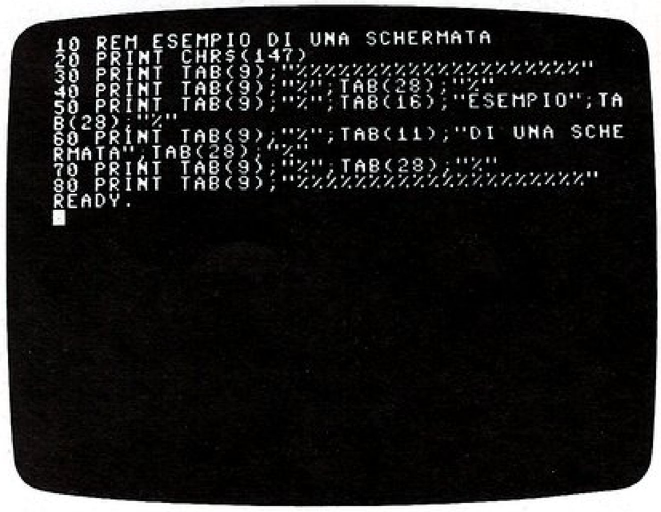


Mentre inserite il programma, noterete che, ora, i comandi non sono più eseguiti quando premete RETURN: il programma è depositato nella memoria del computer, ed è sufficiente eseguirlo (immettendo RUN e premendo RETURN) per vedere cosa produce sullo schermo.

Come numerare le righe di un programma

Potreste essere stupiti dal fatto che le righe sono numerate per decine. Quando state scrivendo e provando dei programmi, capiterà frequentemente di dover aggiungere righe in più; se le righe fossero numerate, 1, 2, 3, 4 e così via, non ci sarebbe posto per immettere le nuove righe in modo che il computer possa eseguirle nell'ordine corretto. Nel programma precedente c'è invece spazio per aggiungere, se necessario, altre righe numerate da 0 a 9, da 11 a 19 e così via. Il programma è ancora immagazzinato in memoria, perciò, per provare il prossimo, spegnete e riaccendete il computer per reinizializzarlo e cancellare il vecchio programma. Guardate, ora, che cosa produce il prossimo programma dopo essere stato immesso e dopo aver battuto RUN:

PROGRAMMA "IMMAGINE VIDEO"



Innanzi tutto, che cosa è un REM? REM è l'abbreviazione di REMark (commento); il computer non esegue nulla quando incontra REM, si limita a memorizzarla con il resto del programma, ma essa è un utile mezzo per assegnare dei titoli o dei nomi ai programmi o alle righe, in modo da renderli più comprensibili. Man mano che la vostra abilità di programmatore aumenterà, troverete le righe REM sempre più utili per ricordarvi come funziona un determinato programma; anche altre persone saranno aiutate nella comprensione dei vostri programmi dalla presenza di righe REM che spiegano cosa state facendo o come è strutturato il programma.

PRINT CHR$(147) è un comando che avete già utilizzato: è un modo rapido per togliere tutte le informazioni inutili dallo schermo. Le righe dalla 30 alla 80 costruiscono una cornice di simboli "%" attorno al testo; il programma usa anche il comando TAB per posizionare i simboli su ogni linea: ci sono 40 posizioni per i caratteri su ogni riga video, numerate da 0 a 39. TAB funziona come il tabulatore di una macchina per scrivere, perciò PRINT TAB(9) significa che la visualizzazione del testo deve cominciare alla posizione 9, invece che dal margine sinistro dello schermo. Le righe 50 e 60 visualizzano il testo stesso al centro della cornice di "%".

Perché la punteggiatura è importante

Il prossimo programma utilizza le tecniche mostrate alle pagg. 18 e 19; spegnete ancora il computer per un secondo o due prima di immetterlo:

PROGRAMMA "CALCOLATRICE"

```
10 PRINT CHR$(147)
20 PRINT : PRINT TAB(8);"43/3 =";43/3
30 PRINT : PRINT TAB(8);"3*18 =";3*18
40 PRINT : PRINT TAB(8);"6.25*4 =";6.25*
4
50 PRINT : PRINT TAB(8);"179*9.8 =";179*
9.8
60 PRINT : PRINT TAB(8);"1002/28.7+3 =";
1002/28.7+3
70 PRINT : PRINT TAB(8);"100*9/5+32 ="10
0*9/5+32
█
```

```
        43/3 = 14.3333333

        3*18 = 54

        6.25*4 = 25

        179*9.8 = 1754.2

        1002/28.7+3 = 37.912892

        100*9/5+32 = 212
READY.
█
```

In ognuno dei sei casi lo schermo mostra l'espressione e il risultato di un calcolo. Il comando PRINT all'inizio di ogni riga di programma produce una riga vuota prima che il calcolo successivo venga visualizzato.
Il punto e virgola (";") è molto importante: assicura che l'espressione del calcolo e, immediatamente dopo, il suo risultato, siano mostrati sulla stessa riga dello schermo. Provate ad eseguire lo stesso programma con una virgola, un ":" o uno spazio al posto del punto e virgola: vi renderete immediatamente conto dell'importanza della punteggiatura nella programmazione del computer.
Anche una corretta spaziatura è vitale per ottenere delle visualizzazioni leggibili quando il computer stampa numeri e stringhe sulla stessa riga.
Il seguente programma, che esegue ancora alcuni calcoli con il comando PRINT, vi mostrerà come organizzare una schermata; esso visualizza sullo schermo alcune equivalenze:

PROGRAMMA "EQUIVALENZE"

```
LIST

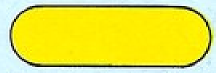
10 PRINT CHR$(147)
20 PRINT "EQUIVALENZE"
30 PRINT "___________"
40 PRINT
50 PRINT "1 PIEDE =";12*2.54;"CENTIMETRI
"
60 PRINT
70 PRINT "18 CENTIGRADI =";(18*9/5)+32;"
FAHRENHEIT"
80 PRINT
90 PRINT "10 CHILOMETRI =";10*5/8;"MIGLI
A"
100 PRINT
110 PRINT "1 ANNO =";365*24*60*60;"SECON
DI"
READY.
█
```

Come scrivere un flowchart

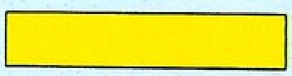
Se un programma funziona correttamente deve effettuare le corrette operazioni nel giusto ordine. Disegnare un flowchart (diagramma di flusso) è un modo utile per pianificare i vari passi necessari perché il computer esegua un particolare compito. Il diagramma di flusso seguente mostra come concepire un programma che addiziona tutti i numeri compresi tra 1 e 1000. Ogni simbolo rappresenta un'operazione diversa, e le frecce che collegano i vari simboli mostrano il percorso che il programma deve seguire. NUMERO e TOTALE possono essere introdotti nel computer come variabili numeriche. Questo programma possiede caratteristiche che incontrerete ancora: un "ciclo" e un "nodo di decisione" il quale determina quante volte deve essere eseguito il ciclo:

COSTRUIRE UN FLOWCHART

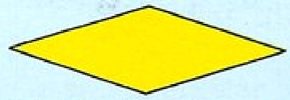
Questo diagramma mostra tutti i passi necessari per realizzare un programma che somma tutti i numeri da 1 a 1000.



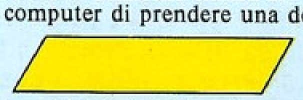
Terminatore Indica l'inizio e la fine del flowchart.



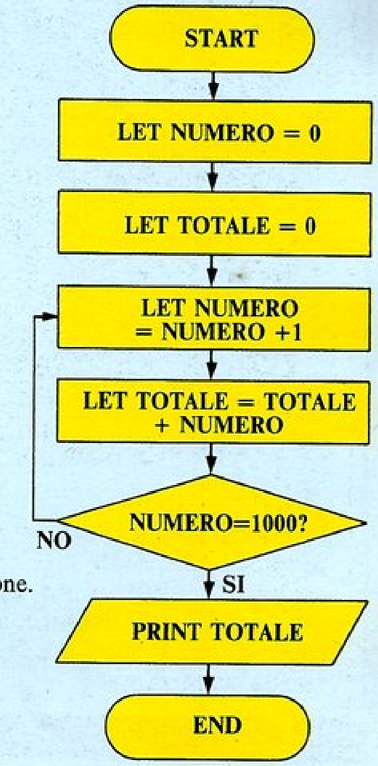
Istruzione Indica ogni differente istruzione.



Nodo di decisione Indica al computer di prendere una decisione.



Input/Output Indica al computer di acquisire o di emettere informazioni.



COME VISUALIZZARE IL TESTO DEI PROGRAMMI

Appena inizierete a scrivere programmi sul computer, avrete spesso bisogno di rivedere parti già inserite per controllare qualcosa o per modificarle in qualche modo. Per fare questo, dovete essere in grado di rivedere il programma sul video anche dopo averlo eseguito. Il Commodore vi permette di rivedere qualsiasi cosa abbiate caricato in memoria: in questo modo potete controllare il "listato" del programma, cioè il programma così come è stato inserito da tastiera.
Dopo aver spento il computer per un attimo, immettete un programma preso da una delle pagine precedenti ed eseguitelo con il comando RUN, per essere sicuri che sia corretto. Il comando basic LIST è usato per richiamare il vostro programma sullo schermo dalla zona di memoria in cui è attualmente memorizzato. Ecco questo comando utilizzato con il primo programma di pag. 20; il programma è stato prima inserito da tastiera, quindi è stato immesso il comando LIST:

RIVEDERE UN PROGRAMMA CON **LIST**

```
10 PRINT CHR$(147)
20 LET X=150
30 LET TEXT$="AL QUADRATO = "
40 PRINT : PRINT " **********************
******************"
50 PRINT : PRINT TAB(8);X;TEXT$;X↑2
60 PRINT : PRINT " **********************
******************"
READY.

LIST

10 PRINT CHR$(147)
20 LET X=150
30 LET TEXT$="AL QUADRATO = "
40 PRINT : PRINT " **********************
******************"
50 PRINT : PRINT TAB(8);X;TEXT$;X↑2
60 PRINT : PRINT " **********************
******************"
READY.
█
```

Ogni volta che premete RETURN dopo aver scritto una riga di programma, quest'ultima è memorizzata nella RAM: il listato del programma è la copia esatta di tutte le righe che la RAM contiene attualmente. Il programma non è fisicamente trasferito dalla RAM allo schermo da LIST, ma rimane caricato in memoria.

Come spostarsi attraverso il testo

La possibilità del comando LIST non finiscono qui. Inserite il programma mostrato nella prossima schermata: esso utilizza una tecnica che non avete ancora imparato, ma ciò non ha importanza (incidentalmente, se volete eseguire questo programma, premete il tasto RETURN dopo aver immesso il vostro nome); poi, per visualizzare ancora l'intero programma, battete LIST. Supponiamo che vogliate rivedere solo la prima riga del programma: potete fare ciò immettendo LIST 10. Forse, però, vi interessano solo poche righe all'interno del programma, per esempio quelle da 20 a 40. Provate ad immettere LIST 20-40. Potete vederne il risultato nella seconda schermata:

PROGRAMMA "OPERATORE"

```
10 PRINT CHR$(147)
20 PRINT : PRINT TAB(5);"----------------
----------------"
30 PRINT : PRINT TAB(7);"QUALE E' IL TUO
 NOME"; : INPUT NOME$
40 PRINT : PRINT TAB(2);"COMMODORE 64 -
PROGRAMMATO DA ";NOME$
50 PRINT : PRINT TAB(5);"----------------
----------------"
█
```

LISTATO PARZIALE

```
10 PRINT CHR$(147)
20 PRINT : PRINT TAB(5);"----------------
----------------"
30 PRINT : PRINT TAB(7);"QUALE E' IL TUO
 NOME"; : INPUT NOME$
40 PRINT : PRINT TAB(2);"COMMODORE 64 -
PROGRAMMATO DA ";NOME$
50 PRINT : PRINT TAB(5);"----------------
----------------"

LIST 20-40

20 PRINT : PRINT TAB(5);"----------------
----------------"
30 PRINT : PRINT TAB(7);"QUALE E' IL TUO
 NOME"; : INPUT NOME$
40 PRINT : PRINT TAB(2);"COMMODORE 64 -
PROGRAMMATO DA ";NOME$

READY.
█
```

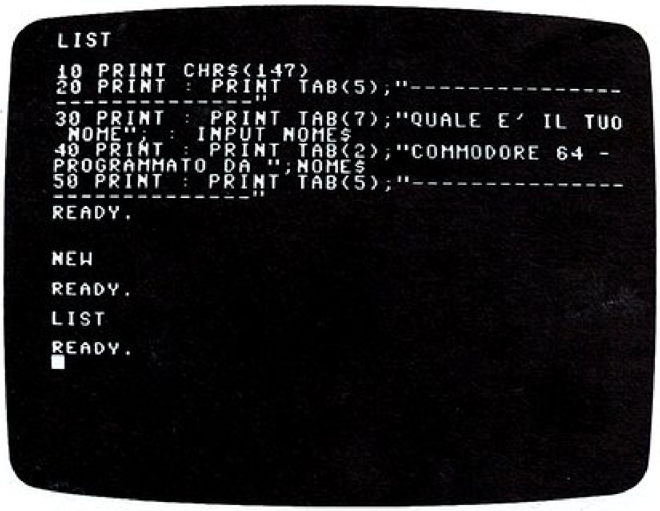
LIST permette anche di vedere ogni cosa fino ad un certo numero di riga, o da una certa riga fino alla fine del programma. Immettete LIST -50 e LIST 20- e controllate il risultato nei due casi.
Se volete cercare una particolare riga (o più di una) in un programma di grosse dimensioni, il metodo migliore è quello di utilizzare LIST senza numeri di riga, ottenendo di far scorrere l'intero programma sullo schermo. Il programma scorrerà rapidamente ma, quando sarete vicino alla parte del listato contenente la riga che state cercando di trovare, potete rallentarne lo scorrimento mantenendo premuto il tasto CTRL: questo vi permetterà di avere più tempo per individuare la riga cercata. Quando l'avrete trovata, il comando LIST potrà essere interrotto premendo il tasto STOP; potrete quindi visualizzare quella particolare riga per poterla esaminare e, se necessario, potrete modificarla utilizzando le tecniche illustrate alle pagg. 24 e 25.

Come usare NEW per cancellare un programma
Immaginate di iniziare a scrivere un nuovo programma; cancellate lo schermo con SHIFT e CLR e quindi immettete la prima riga:

10 PRINT "PROGRAMMA SONDA SPAZIALE"

poi battete RUN. Accadrà qualcosa di strano: l'ultimo programma è tutt'ora in memoria. Il computer esegue sì la vostra nuova riga 10, ma poi continua ad eseguire il vecchio programma, poiché non lo avete cancellato dalla memoria. Fino ad ora avete spento e poi riacceso la macchina prima di immettere un nuovo programma, ma ci sono modi migliori per disfarsi di quelli vecchi. Uno di questi consiste nell'uso del comando BASIC NEW. Immettete NEW, premete il tasto RETURN, quindi reinserite la riga 10: questa volta il vecchio programma è stato eliminato; se provate a listare un qualsiasi programma dopo aver immesso NEW, scoprirete che tutte le righe memorizzate sono scomparse:

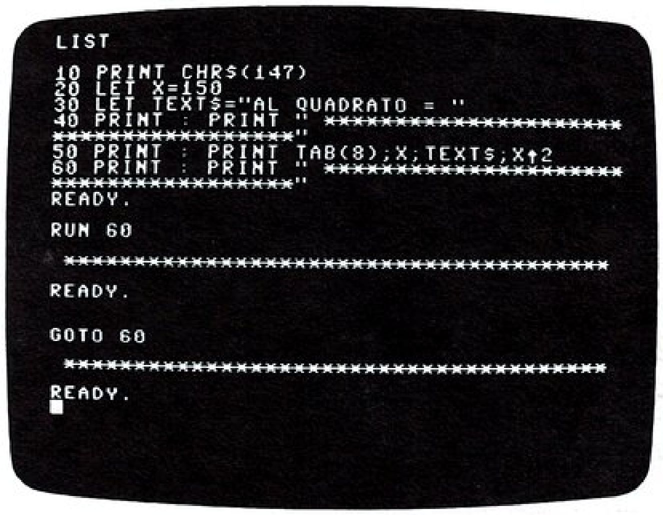
COME USARE **NEW**



Prima di usare NEW, dovete però essere sicuri di voler cancellare veramente il programma in memoria; in effetti c'è un modo per recuperare un programma cancellato con NEW (è mostrato a pag. 57), ma è un metodo poco pratico e, inoltre, funziona solo se non avete già iniziato ad immettere un nuovo programma.

Come eseguire una parte di un programma
I programmi possono essere eseguiti parzialmente usando RUN seguito da un numero di riga, oppure con il comando GOTO.
Potreste avere dei problemi con una parte del programma che non funziona correttamente; se il programma è corto, è più conveniente provare l'intero programma piuttosto che una parte di esso, ma cosa fare se la parte che volete provare ad eseguire e rieseguire si trova nella parte finale di un programma molto lungo?
Diventa piuttosto noioso, ed è una perdita di tempo, dover aspettare magari cinque minuti (durante i quali tutto il resto del programma viene eseguito) prima di poter controllare la parte sospetta che si trova vicino alla fine. Supponiamo che, nel programma di calcolo dei quadrati di pag. 20, vogliate controllare il funzionamento dalla riga 60 in poi: invece di usare solo RUN, provate ad immettere RUN 60 oppure GOTO 60 e provate a vedere che cosa succede:

PROGRAMMA ESEGUITO PARZIALMENTE

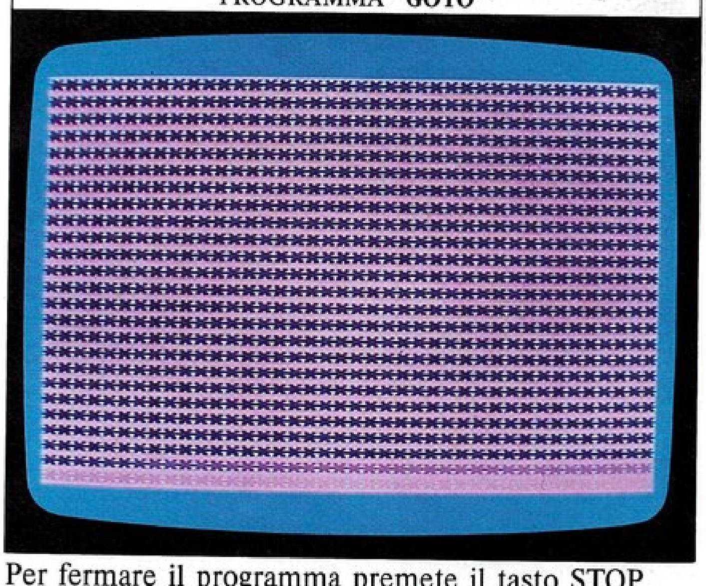


GOTO è uno dei più semplici e più utili comandi del linguaggio BASIC. Usato in questo caso (senza un numero di riga che lo preceda), GOTO permette al computer di saltare ad una specifica riga e quindi di eseguire il programma da quel punto. Quando GOTO è inserito in una riga di programma, invece, il risultato diventa molto interessante.
Potete avere un'idea di cosa può fare questo comando inserendo semplicemente queste due righe di programma:

10 PRINT "*";
20 GOTO 10

Quando le eseguirete vi apparirà un'immagine simile alla seguente (a seconda di quali colori avete selezionato):

PROGRAMMA **"GOTO"**



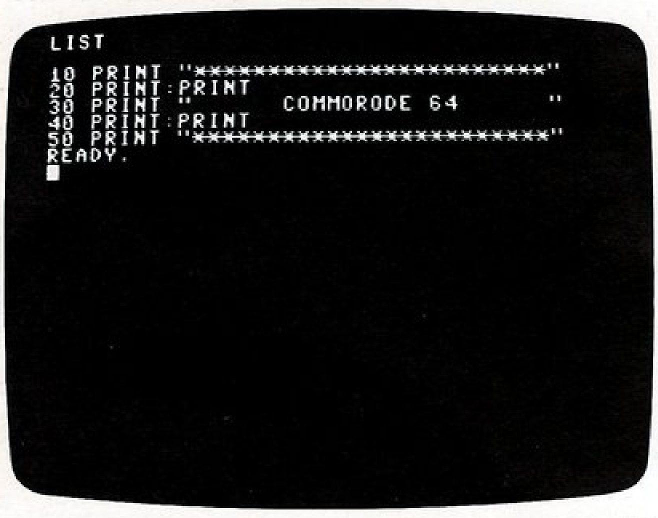
Per fermare il programma premete il tasto STOP.
Se non riuscite a capire come mai appare questa immagine non preoccupatevi: si riparlerà dei comando GOTO dopo che avrete acquistato familiarità con le istruzioni del BASIC.

COME CORREGGERE GLI ERRORI

La programmazione dei computer è un passatempo in cui gli errori sono all'ordine del giorno. I programmi molto raramente funzionano in modo soddisfacente al primo tentativo, e più sono lunghi, più è difficile farli funzionare correttamente. È importante che vi rendiate conto che fare degli errori e correggerli è spesso una parte interessante dello sviluppo dei programmi, perciò non nascondete e non ignorate i vostri errori: essi sono indispensabili per imparare a fare le cose nel modo giusto. Per esempio, in un programma non potete cambiare la punteggiatura senza cambiare completamente il senso di quello che avete scritto: come è stato mostrato a pag. 21, questo fatto potrebbe produrre dei cambiamenti. Per il computer, la punteggiatura ha un significato molto preciso, e se voi la sbagliate, un programma non potrà mai funzionare.
Potete cambiare una riga in un programma in due modi. Primo, potete reinserirla e battere RETURN: la nuova versione automaticamente rimpiazza la vecchia nella memoria del computer. Tuttavia quando si tratta di un piccolo errore all'interno di una riga molto lunga, è una perdita di tempo reinserirla completamente. Un metodo alternativo per operare dei cambiamenti in questi casi è quello di usare i tasti di controllo del cursore.

Come effettuare modifiche sullo schermo (Editing)
L'Editing necessita dell'uso dei due tasti CRSR su cui sono stampate due frecce e del tasto INST/DEL. Ecco un programma che richiede di essere modificato:

IL PROGRAMMA DA MODIFICARE

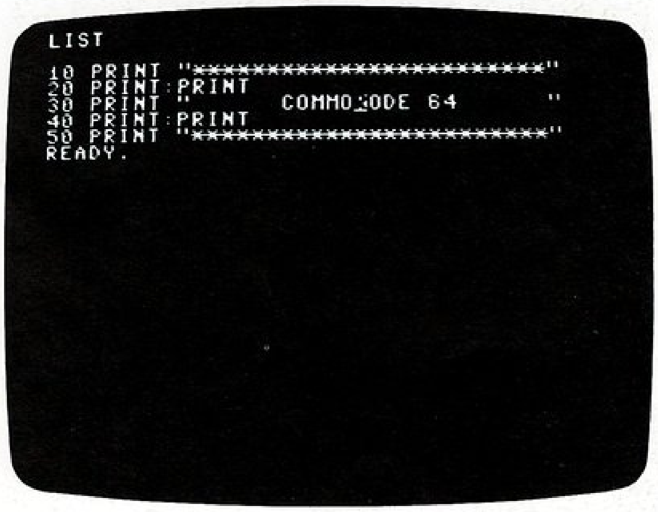


Per correggere l'ortografia della riga 30, così da leggere:

30 PRINT "COMMODORE 64"

dovreste ribattere tutta la riga. Provate ad usare, invece, l'editor di schermo.
Per prima cosa, immettete il programma ed eseguitelo, visualizzatelo quindi sullo schermo. Premete SHIFT insieme con il tasto CRSR con le frecce verso l'alto e verso il basso: il cursore lampeggiante si sposterà di una riga verso l'alto. Se mantenete premuti questi tasti il movimento sarà ripetuto finché essi non saranno rilasciati.
Se premete il tasto CRSR sopra/sotto senza premere il tasto SHIFT, il cursore lampeggiante sullo schermo si muoverà in basso di una riga: anche questo movimento sarà ripetuto se manterrete premuti questi tasti.
Usando questi mezzi dovrete muovere il cursore sullo schermo fino a sovrapporlo al carattere "3" della riga 30 (la riga che deve essere modificata). Ora, usando il tasto CRSR sinistra/destra, muovete il cursore lampeggiante lungo la riga 30 fino a sovrapporlo alla lettera "R" in COMMORODE. Potete così immettere la lettera corretta e questa, automaticamente, rimpiazzerà la precedente sullo schermo. Dopo di che premete RETURN.
Potete anche aggiungere spazi extra in una riga, per poter inserire lettere che avete dimenticato. Per aggiungere uno spazio extra dovete premere il tasto SHIFT insieme a quello indicato INST/DEL in alto a destra sulla tastiera: se lo fate, le lettere RODE e le virgolette di chiusura alla fine della riga saranno spostate di una posizione a destra e potete introdurre un altro carattere. Per dire al Commodore che avete finito di modificare una riga, dovete premere RETURN con il cursore lampeggiante posto da qualche parte sulla riga modificata. Se non premete RETURN il computer non potrà memorizzare nessuno dei vostri cambiamenti al programma:

IL PROGRAMMA DURANTE L'EDITING



Vorrete spesso aggiungere righe ad un programma dopo che lo avrete scritto in una prima versione: potreste, per esempio, esservi scordati di mettere PRINT CHR$(147) all'inizio, per incominciare il programma con lo schermo cancellato. Non dovete modificare un qualche numero di riga per fare ciò; nel programma precedente, per esempio, potete introdurre la nuova prima riga immettendo:

5 PRINT CHR$(147)

Poiché il computer esegue le istruzioni BASIC seguendo l'ordine dei numeri di riga, questa sarà eseguita per prima.

Primi passi nella caccia agli errori

Gli errori nei programmi sono detti "bachi" (Bug), e l'attività di eliminazione degli stessi, debugging. Come avete probabilmente scoperto, il Commodore aiuta molto nel debugging dei programmi, esaminando ciò che avete inserito per cercare errori nell'ortografia, o nella grammatica, o nella sintassi. Se il computer ne trova uno, vi avverte visualizzando un messaggio di errore sullo schermo; avete già visto i messaggi BAD SUBSCRIPT ERROR e SYNTAX ERROR a pag. 14.

Il Commodore può visualizzare 28 differenti messaggi d'errore: potreste aver avuto già a che fare con qualcuno di loro se avete commesso qualche errore nell'immettere un programma.

In molti casi i messaggi di errore vi diranno a che riga del vostro programma sono stati riscontrati gli errori; per correggere un errore, allora, visualizzate la parte interessata del programma e modificatela usando i tasti CRSR e INST/DEL. Ecco alcuni programmi che non vogliono funzionare. Provate ad eseguirli e quindi controllate i messaggi di errore che essi producono nella tabella che segue. Dovreste essere in grado di individuare quale riga è la causa del problema in ognuno dei programmi:

PROGRAMMI ERRATI

```
LIST

10 FOR X=60 TO 100
20 PRINT 2↑X,3↑X
30 NEXT X
READY.
```

```
LIST

10 FOR X=0 TO 20
20 INPUT B
30 PRINT B/X
40 NEXT X
READY.
```

PROGRAMMA ERRATO

```
LIST

10 INPUT "BATTI UN NUMERO ";A
20 FOR F=1 TO 12
30 PRINT TAB(9);F;"*";A;"=";F*A
40 NEXT G
READY.
```

MESSAGGI DI ERRORE DEL COMMODORE

Ecco alcuni messaggi di errore che potete incontrare quando scrivete i vostri primi programmi.

Bad subscript
Questo messaggio si riferisce in particolare all'indice di un elemento di un "array" il cui valore non è ammesso, ma potete pensare che la sua causa sia semplicemente un errore di battitura (guardate a pag. 14).

Break
Un programma è stato interrotto perché avete premuto il tasto RUN/STOP. Il messaggio vi indicherà dove il programma è stato interrotto.

Device Not Present
Questo messaggio compare quando tentate di salvare o di recuperare un programma senza aver connesso il registratore a cassette o il drive per il disco (guardare a pag. 58). Apparirà anche se non avete acceso il drive!

Division by Zero
Avete chiesto al computer di eseguire un calcolo che include una divisione per zero: è una operazione matematica impossibile.

Formula too Complex
Questo messaggio compare se avete scritto un'espressione con troppe parentesi perché il computer la possa calcolare.

Illegal Quantity
Un numero che avete utilizzato con la funzione aveva un valore esterno all'insieme di quelli possibili: SQR (radice quadrata) di un numero negativo, per esempio.

NEXT without FOR
Questo messaggio compare quando un ciclo FOR...NEXT usa, per errore, due variabili di controllo anziché una, o quando più cicli FOR...NEXT sono innestati in modo errato (guardate alle pagg. 28 e 56,57).

Overflow
Il risultato di un calcolo è troppo grande (guardate a pag. 19).

Redo from Start
In un programma che utilizza l'istruzione INPUT (guardate alle pagg. 26 e 27), avete immesso una lettera quando il computer si aspettava un numero. Dopo questo messaggio il computer aspetta che immettiate finalmente il numero.

String too Long
Avete programmato il computer per creare una stringa con un numero di caratteri superiore al valore massimo consentito (255).

?Syntax error
Il computer non è in grado di interpretare il messaggio che avete immesso.

Type Mismatch
Avete mischiato lettere e numeri (guardate alle pagg. 14 e 15).

CONVERSARE CON IL COMPUTER

In tutti i programmi che avete scritto sino ad ora, avete dato al computer un certo insieme di istruzioni, e avete lasciato che esso le eseguisse. Ogni programma forniva un solo risultato che era esattamente lo stesso ogni volta che il programma era eseguito; in realtà pochi programmi funzionano così. In un video gioco, per esempio, il giocatore fornisce al computer nuove istruzioni ogni volta che il gioco è eseguito; il computer riceve queste informazioni durante lo svolgimento del gioco modificando la visualizzazione in funzione delle stesse. In effetti, è difficile scrivere un programma di qualsiasi complessità che non richieda di essere interrotto durante la sua esecuzione per acquisire nuove informazioni.
Il comando BASIC INPUT è pensato per poter essere usato in questa situazione. Esso vi permette di realizzare una conversazione con il computer: voi "parlate" alla macchina attraverso la tastiera ed essa vi "parla" attraverso lo schermo.
Il comando INPUT permette al computer di ricordare informazioni immesse da tastiera e dare a queste un nome (con una variabile numerica se l'informazione è un numero, con una variabile stringa se l'informazione è alfanumerica). Una volta che il computer ha dato un nome all'informazione, la può passare ad un'altra parte del programma che la utilizzerà in seguito. Ecco un esempio dell'uso di INPUT:

PROGRAMMA "INPUT"

```
LIST

10 PRINT CHR$(147)
20 PRINT "QUALE E' IL TUO NOME"
30 INPUT NOME$
40 PRINT CHR$(147)
50 PRINT "*************************"
60 PRINT "PROGRAMMA DI ";NOME$
70 PRINT "*************************"
READY.
█
```

Domande dal vostro computer
Il programma ordina al computer di visualizzare la domanda "QUALE È IL TUO NOME" e la riga 30 vi aggiunge un punto interrogativo per indicare che il computer è in attesa di nuove informazioni. Non c'è bisogno di affrettarsi, non c'è un limite di tempo. Il computer può aspettare in eterno che voi immettiate l'informazione richiesta, qualunque essa sia. Immettete il vostro nome e premete il tasto RETURN. La riga di INPUT del programma prende il vostro nome e lo contrassegna con la variabile stringa NAME$; il segno "$" indica che il computer è stato programmato per attendersi come risposta una o più lettere.
Questo programma dovrebbe esservi familiare: è simile a quello che è stato illustrato a pag. 22 come un esempio dell'uso del comando LIST. Se lo confrontate con quell'esempio, noterete che INPUT esegue il lavoro fatto da due righe in questo primo programma INPUT: esso fa in modo che sia visualizzata la domanda e qui interrompe il programma in attesa della vostra risposta; inoltre visualizza automaticamente un punto di domanda (perciò è inutile che lo aggiungiate voi). Le prossime due schermate mostrano come INPUT può funzionare in modo analogo al comando PRINT. Notate il punto e virgola che compare prima di NAME$ alla riga 20:

USARE INPUT PER VISUALIZZARE LA DOMANDA

```
LIST

10 PRINT CHR$(147)
20 INPUT "QUALE E' IL TUO NOME";NOME$
30 PRINT CHR$(147)
40 PRINT "*************************"
50 PRINT "PROGRAMMA DI ";NOME$
60 PRINT "*************************"
READY.
█
```

```
*************************
PROGRAMMA DI GIANNI
*************************

READY.
█
```

Come usare INPUT per acquisire dati numerici
Potete anche usare INPUT per acquisire dati numerici mentre il programma è in esecuzione. Questo fatto ha molte applicazioni pratiche.

Considerate, per esempio, il problema di convertire lunghezze, dimensioni o pesi da una unità di misura in un'altra. La conversione è sempre la stessa (2.54 centimetri per pollice, 0.3048 metri per piede, 2.2 libbre per chilogrammo e così via), ma i numeri sono di volta in volta differenti.
Ecco un semplice programma di conversione che potete provare:

PROGRAMMA EQUIVALENZE

```
LIST

10 PRINT CHR$(147)
20 PRINT "PROGRAMMA DI CONVERSIONE"
30 PRINT : INPUT "QUANTE MIGLIA";M
40 PRINT : PRINT M;"MIGLIA = ";M*1.61;"C
HILOMETRI"
READY.
█
```

Il programma vi chiede quante miglia volete convertire in chilometri, aspetta la vostra risposta, esegue il calcolo e, quindi, visualizza il risultato sullo schermo. Poiché la riga di INPUT si aspetta un numero in risposta alla domanda, la variabile che crea è il tipo numerico.

Formato degli OUTPUT
Potete osservare che il risultato del programma precedente compare su una singola linea, con i vari numeri e stringhe visualizzati gli uni accanto alle altre.
Provate a modificare la riga 40 per cambiare il punto e virgola in virgola:

VISUALIZZAZIONE MODIFICATA

```
PROGRAMMA DI CONVERSIONE

QUANTE MIGLIA? 12

 12        MIGLIA =    19.32     CHILOMETRI

READY.

LIST

10 PRINT CHR$(147)
20 PRINT "PROGRAMMA DI CONVERSIONE"
30 PRINT : INPUT "QUANTE MIGLIA";M
40 PRINT : PRINT M,"MIGLIA = ",M*1.61,"C
HILOMETRI"
READY.
█
```

Ora lo schermo è diviso in quattro zone invisibili (colonne); ognuna di queste è larga 10 caratteri.
Usando la virgola nell'istruzione PRINT per separare i dati che devono essere visualizzati, ogni termine compare all'inizio di ogni colonna. Immettendo il seguente programma di esempio, questo concetto vi apparirà chiaro:

COLONNE DI DIVISIONE DELLO SCHERMO

```
PRINT 1;2;3;4;5;6;7;8
 1  2  3  4  5  6  7  8

READY.


PRINT 1,2,3,4,5,6,7,8
 1         2         3         4
 5         6         7         8
READY.
█
```

Ulteriori informazioni su TAB
Benché il computer posizionerà automaticamente sia numeri che stringhe in zone predefinite dello schermo, non siete costretti a visualizzare tutti i risultati in questo modo. In realtà, come avete visto a pag. 20, potete visualizzare un dato in una posizione qualsiasi di una linea usando la funzione BASIC TAB. Questa funzione è sempre seguita da un numero tra parentesi che determina dove deve cominciare la visualizzazione:

PRINT TAB(2); "TAB 2"

visualizza la stringa "TAB 2" a partire dalla seconda posizione da sinistra. Ecco ancora vari esempi di uso di TAB:

COME USARE TAB

```
PRINT TAB(5);"TAB (5)"
     TAB (5)
READY.
PRINT TAB(15);"TAB (15)"
               TAB (15)
READY.
PRINT TAB(25);"TAB (25)"
                         TAB (25)
READY.
PRINT TAB(30);"TAB (30)"
                              TAB (30)
READY.
█
```

COME INSERIRE DEI CICLI IN UN PROGRAMMA

I computer sono molto adatti ad eseguire velocemente un gran numero di operazioni semplici e ripetitive. Se si tratta però di operazioni che comportano numerose ripetizioni, un computer deve poter eseguire lo stesso programma o parte di un programma più di una volta. A pag. 23 avete realizzato un ciclo usando un GOTO; lo stesso comando è usato ora in un ciclo più complesso (la riga 10 imposta semplicemente i colori):

PROGRAMMA "CICLO SENZA FINE"

```
LIST

10 POKE 53280,6:POKE 53281,7
20 PRINT CHR$(144);CHR$(147)
30 LET X=1
40 PRINT X,X*2,X*X
50 LET X=X+1
60 GOTO 40
READY.
```

Se eseguite questo programma, scoprirete lo svantaggio dell'uso del solo GOTO: il programma non termina mai. Premete STOP per fermarlo. Sullo schermo sarà indicato a quale riga il programma si è fermato:

CICLO INTERROTTO

```
 108        216       11664
 109        218       11881
 110        220       12100
 111        222       12321
 112        224       12544
 113        226       12769
 114        228       12996
 115        230       13225
 116        232       13456
 117        234       13689
 118        236       13924
 119        238       14161
 120        240       14400
 121        242       14641
 122        244       14884
 123        246       15129
 124        248       15376
 125        250       15625
 126        252       15876
 127        254       16129
 128        256       16384

BREAK IN 40
READY.
```

Come uscire da un ciclo

La soluzione per questi programmi senza fine è il ciclo FOR...NEXT: esso vi permette di fissare quante volte il ciclo deve essere eseguito. Potete usarlo per stampare la stessa tabella del primo programma:

PROGRAMMA "CICLO **FOR...NEXT**"

```
LIST

10 POKE 53280,6:POKE 53281,7
20 PRINT CHR$(144);CHR$(147)
30 FOR X=1 TO 20
40 PRINT X,X*2,X*X
50 NEXT X
READY.
```

Il ciclo FOR...NEXT migliora il programma e lo accorcia di una riga; notate che ora non avete dovuto porre X = 1, poiché FOR...NEXT provvede a questo automaticamente.

Il ciclo inizia alla riga 30 con X = 1; la riga 50 imposta il successivo valore di X e il programma riparte dalla riga 30. Questo ciclo continua finché X non assume il valore 20, il massimo fissato dalla riga 30.

In questo caso, il programma si ferma poiché la 50 è l'ultima riga di programma. Se necessario, il ciclo può essere interrotto ad ogni passata per aspettare nuove informazioni. Provate ad usare INPUT in un ciclo FOR...NEXT:

PROGRAMMA **"FOR...NEXT/INPUT"**

```
LIST

10 FOR N=1 TO 5
20 PRINT CHR$(147)
30 PRINT : PRINT TAB(6);"CONVERSIONE DI
TEMPERATURA"
40 POKE 214,10 : PRINT : INPUT "DAMMI UN
A TEMPERATURA IN FAHRENHEIT";TEMP
45 POKE 214,18 : PRINT : POKE 211,3
50 PRINT TEMP;"FAHRENHEIT = ";INT((TEMP-
32)*5/9);"GRADI CENTIGRADI"
60 FOR A=1 TO 5000
70 NEXT A
80 NEXT N
READY.
```

Questo programma converte una temperatura espressa in gradi Fahrenheit, nella corrispondente temperatura in gradi centigradi.

Il ciclo FOR...NEXT inizia alla riga 10 fissando un limite di cinque conversioni dopo le quali, se volete, potete

nuovamente eseguire il programma. L'istruzione INPUT alla riga 40 sospende il programma finché voi non immettete la temperatura Fahrenheit che volete convertire. La riga 50 esegue il calcolo e stampa il risultato.

Come rallentare i vostri programmi
Potreste essere rimasti stupiti dalle righe 60 e 70. Queste servono per evitare che il computer stampi e cancelli i risultati prima che voi facciate in tempo a leggerli: si tratta cioè di un ritardo che mantiene ogni risultato sullo schermo per alcuni secondi prima di proseguire. Questo ciclo non fa altro che ritardare l'esecuzione del resto del programma ed è normalmente scritto come una riga singola:

```
60 FOR A=1 TO 5000: NEXT A
```

Potete perfino omettere la A finale: in questo caso il Commodore capirà che il NEXT si riferisce al FOR che lo precede. In generale i due punti possono essere usati in questo modo per separare più comandi su una singola riga, invece che scriverli su più righe. L'inserire un ciclo dentro ad un altro, come in questo caso, è una tecnica detta "innestamento" (nesting) dei cicli. Quando innestate dei cicli nei vostri programmi, assicuratevi che ci sia un NEXT per ogni FOR.

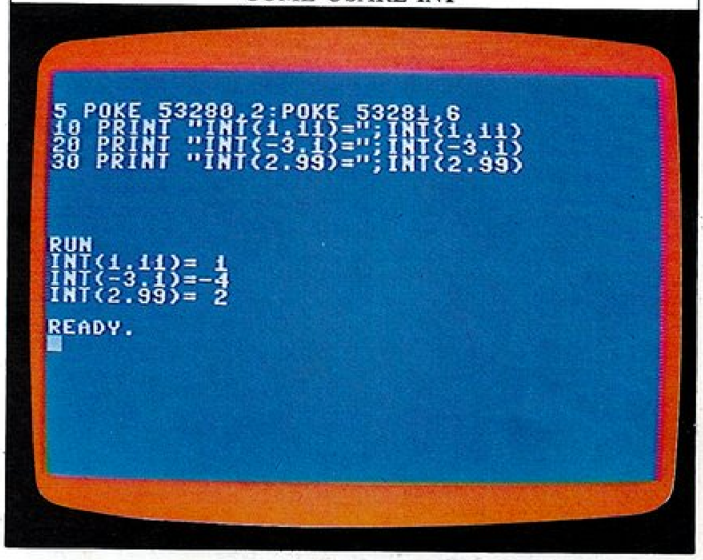
Come arrotondare i numeri
Quando eseguite il programma di conversione, provate ad inserire una temperatura di 32 Fahrenheit. Il programma vi dirà che questa temperatura equivale a 0 gradi centigradi, come vi sareste aspettati. La volta successiva immettete il valore 34; il computer ora vi dirà che 34 gradi Fahrenheit equivalgono a 1.11111111 gradi centigradi, spezzando la parola centigradi su due linee e rendendo confusa la schermata. C'è però un metodo per risolvere questo problema: esso è sufficientemente preciso per la maggior parte degli scopi se le risposte sono costituite da un numero intero di gradi che non occupi molto spazio sulla linea. Provate ad usare l'editor di schermo per sostituire l'espressione (TEMP-32)*5/9 alla riga 50 con l'espressione più complessa INT((TEMP-32)*5/9). Scoprirete che questa operazione ha risolto il problema:

UNA CONVERSIONE ARROTONDATA



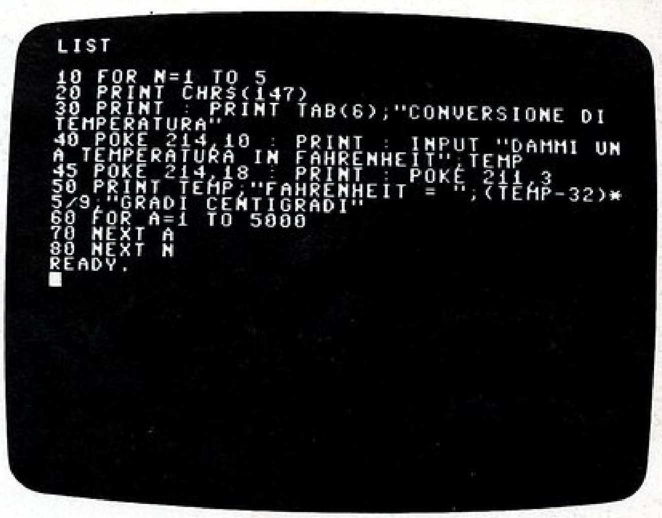
Il numero si presenta meglio e il complesso è molto più gradevole. INT, abbreviazione di INTeger, converte un numero decimale in un numero intero. Se il risultato è, per esempio, 1.11111111, aggiungendo INT questo viene modificato in 1, una approssimazione che è sufficientemente accurata per la maggior parte degli scopi. Quando usate INT, ricordatevi però che esso arrotonda i numeri per difetto; questo può creare dei problemi con numeri che hanno una parte decimale prossima all'unità, come mostra il programma seguente:

COME USARE INT



Il primo esempio è simile ad uno che avete già visto mentre il secondo può sembrare un po' sconcertante, a meno che non vi ricordiate che -4 è minore di -3.1 e che INT arrotonda all'intero inferiore.
L'ultimo esempio fornisce la risposta che vi aspettate, ma riguardando il programma di conversione, sarebbe meglio se 2.99 fosse arrotondato a 3 invece che a 2. Questa si può fare modificando ancora la riga 50:

IL DEFINITIVO PROGRAMMA DI CONVERSIONE



Addizionando 0.5 alla temperatura potete essere sicuri che, in ogni conversione, INT produrrà il numero intero più vicino.

PROGRAMMARE CON I NODI DI DECISIONE

Nelle due pagine precedenti avete visto come i cicli possano essere usati per fare in modo che il programma esegua per un certo numero di volte la stessa sequenza di comandi. Se volete realizzare un calcolo oppure visualizzare qualcosa sullo schermo per 10 volte, per esempio, potreste inserire:

```
FOR A=1 TO 10...
NEXT A
```

Ma c'è un altro modo di farlo, cioè usando una istruzione IF...THEN. Supponiamo che vogliate visualizzare in una tabella tutti i numeri da 1 a 10 insieme con i loro quadrati e i loro cubi. Per prima cosa, ecco cosa potete fare con un ciclo FOR...NEXT:

CICLO **FOR...NEXT**

```
LIST

10 POKE 53280,2:POKE 53281,2
20 PRINT CHR$(147);CHR$(5)
30 PRINT " A"," A↑2"," A↑3"
40 PRINT "-----------------------------" :
 PRINT
50 FOR N=1 TO 10
60 PRINT N,N*N,N*N*N
70 NEXT N
READY.
█
```

```
 A          A↑2         A↑3
---------------------------
 1           1           1
 2           4           8
 3           9           27
 4           16          64
 5           25          125
 6           36          216
 7           49          343
 8           64          512
 9           81          729
 10          100         1000

READY.
█
```

Nel programma con IF...THEN che segue, la riga 10 fissa il colore dello schermo e la riga 30 visualizza l'intestazione della tabella. La 60 è la prima riga del ciclo, essa incrementa N di 1 ad ogni passata. La riga 40 è la stessa istruzione di visualizzazione usata nel programma con FOR...NEXT. Nella riga 80 il computer prende una decisione esaminando N; il simbolo < è l'abbreviazione usata dai matematici per "minore di". Così, se N è minore di 10, il computer riprende il programma dalla riga 60 (notate che la parola chiave GOTO può essere omessa). Quando N diventa uguale a 10 il programma termina:

CICLO **IF...THEN**

```
LIST

10 POKE 53280,2:POKE 53281,2
20 PRINT CHR$(147);CHR$(5)
30 PRINT " A"," A↑2"," A↑3"
40 PRINT "-----------------------------" :
 PRINT
50 N=0
60 N=N+1
70 PRINT N,N*N,N*N*N
80 IF N<10 THEN 60
READY.
█
```

Perché usare IF...THEN?

Potreste meravigliarvi del fatto che, dopo quanto è stato detto, il ciclo IF...THEN produce solo gli stessi risultati del ciclo FOR...NEXT. L'importanza di IF...THEN è che il computer può rispondere a ogni informazione che inserite tramite INPUT durante il funzionamento del programma, prendendo una decisione in base a essa. Ecco un esempio a questo riguardo, che vi da la possibilità di provare la vostra abilità nei calcoli aritmetici mentali (il nuovo comando RND che compare alla riga 70 è spiegato alle pagg. 52 e 53):

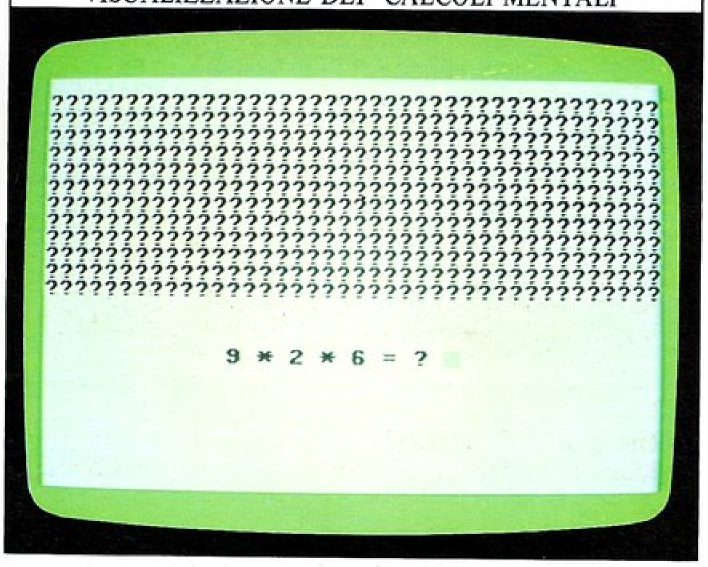
PROGRAMMA "CALCOLI MENTALI"

```
LIST

10 PRINT CHR$(147);CHR$(144): POKE 53280
,5: POKE 53281,15
20 FOR Y=1 TO 12
30 FOR X=1 TO 40
40 PRINT "?";
50 NEXT X : NEXT Y
60 POKE 214,6 : PRINT : POKE 211,13
70 X=INT(RND(0)*10) : Y=INT(RND(0)*10) :
 Z=INT(RND(0)*10)
80 POKE 214,15 : PRINT : POKE 211,11
90 PRINT "                "
100 POKE 214,15 : PRINT : POKE 211,11
110 PRINT X;"*";Y;"*";Z;"= "; : INPUT A
120 IF A=X*Y*Z THEN 160
130 POKE 214,15 : PRINT : POKE 211,11
140 PRINT "      SBAGLIATO
150 FOR T=1 TO 500 : NEXT T : GOTO 80
160 POKE 214,15 : PRINT : POKE 211,11
170 PRINT "      ESATTO
180 FOR T=1 TO 500 : NEXT T : GOTO 70
READY.
█
```

Ogni volta, il computer prepara il problema e aspetta la vostra risposta, da cui dipendono poi due diverse serie di eventi.
Se immettete la risposta corretta, l'istruzione IF...THEN alla riga 120 dice al computer di non saltare alla riga 130, ma alla riga 160, visualizzando il messaggio "ESATTO" e quindi preparando un altro problema.
Se la risposta è errata, il computer salta alla riga 130 per mezzo dell'istruzione IF...THEN, ed entra nella routine di errore. È importante ricordare che deve esserci anche una istruzione nel programma per impedire alla routine di errore di proseguire eseguendo anche la routine di risposta corretta.
Nel nostro caso, si tratta della riga 150 che permette al computer di visualizzare ancora il problema dopo un certo ritardo:

VISUALIZZAZIONE DEI "CALCOLI MENTALI"



Come scegliere la giusta condizione
Quando usate IF...THEN, ricordatevi che esiste una grande varietà di "condizioni" che possono seguire la parola chiave IF in una istruzione. I programmi di queste pagine hanno usato < o =, ma questa è solo una parte dell'insieme completo di simboli che il Commodore usa, come potete vedere nella tabella sotto riportata. Potreste pensare che sia possibile usare IF...THEN solo per comparare un numero od una variabile numerica con un'altra. Non è però così. Lo stesso insieme di condizioni può essere usato con le stringhe. Una riga come:

```
30 IF A$="FRED" THEN...
```

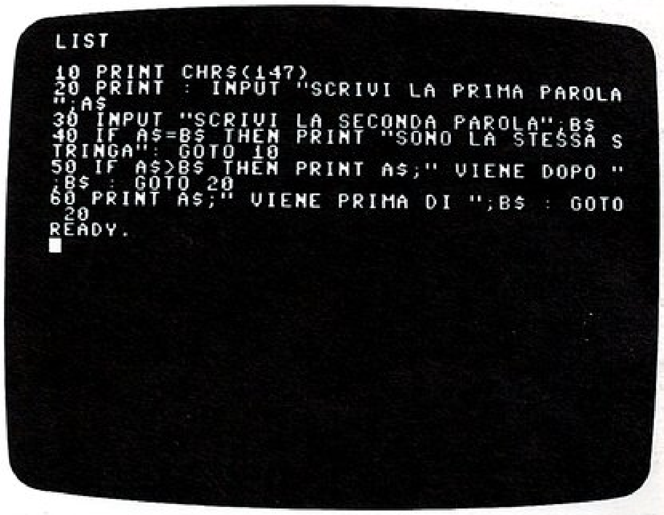
ha almeno un senso compiuto, ma cosa significa la prossima riga?

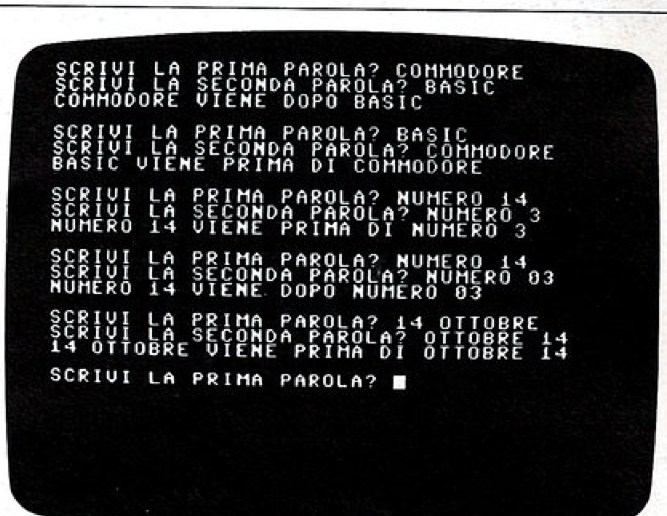
```
30 IF A$>"FRED" THEN...
```

Per poter rispondere, dovete conoscere come il BASIC Commodore confronta due stringhe.

Come ordinare le parole
Quando il Commodore confronta due stringhe, preleva il primo carattere da ognuna di esse ed esamina il relativo codice ASCII. La stringa il cui primo carattere ha il codice ASCII più grande è considerata "maggiore" dell'altra. Ciò significa che FRED sarà maggiore di ALAN, poiché il codice ASCII della F è maggiore di quello della A.
Se i primi caratteri di entrambe le stringhe sono uguali, vengono comparati i secondi caratteri, e così via, fino a quando è rilevata una differenza.
Con due parole come SU ed SUPER, dove tutte le lettere della prima compaiono anche nella seconda, la parola più lunga è considerata maggiore di quella più corta. Solo quando sono composte dalle stesse lettere il computer decide che vale la condizione di uguaglianza.
Se guardate la tabella dei codici ASCII a pag. 61, vedrete che le lettere dell'alfabeto appaiono in ordine, con codici consecutivi da 65 a 90. Questo significa che IF...THEN può essere usato per disporre parole in ordine alfabetico, come mostra questo programma:

ORDINATORE ALFABETICO





CONDIZIONI DI IF...THEN

Questi simboli specificano il tipo di decisione che il computer prenderà riguardo alla condizione che segue IF.

=	è uguale a	<>	non è uguale a
>	è maggiore di	<	è minore di
>=	è maggiore o uguale a	<=	è minore o uguale

POKE E PEEK

Su alcuni microcomputer una grossa quantità dello spazio di memoria è dedicato all'interprete BASIC, in modo che la macchina possa comprendere un grande numero di parole chiave di tale linguaggio. Nel Commodore, invece, il vocabolario del BASIC è piuttosto ridotto e usa una minima parte della memoria, lasciando così spazio per programmi di maggiori dimensioni. Nonostante ciò voi dovete essere in grado di poter programmare il computer per poter produrre disegni e suoni. Entrambe queste possibilità, nelle altre macchine, sono controllate da comandi come COLOR, DRAW e SOUND, nessuno dei quali funziona sul Commodore; grafici, suoni e un vasto insieme di altre funzioni sono invece realizzate per mezzo di due comandi del tutto generali: POKE e PEEK.

Usare POKE per cambiare i colori

Il comando POKE è usato per caricare un valore direttamente in una locazione della RAM del Commodore. Se immettete il programma seguente, potete vedere gli effetti che si ottengono modificando i valori memorizzati in due di questi "indirizzi" di memoria:

PROGRAMMA CHE CAMBIA I COLORI

```
LIST

5 PRINT CHR$(147)
10 FOR C=1 TO 15
20 POKE 53280,C : POKE 53281,C
30 FOR K=1 TO 1000
40 NEXT K
50 NEXT C
READY.
█
```

Quando eseguirete questo programma, vedrete che il bordo e lo sfondo dello schermo passeranno attraverso una sequenza di colori; in effetti il programma utilizza tutti i colori che il Commodore può visualizzare. Il cambiamento vero e proprio dei colori è controllato da due comandi POKE alla riga 20; questa riga dice al computer di caricare (POKE) il valore della variabile C nelle due locazioni di memoria 53280 e 53281. Il Commodore controlla il contenuto di queste due particolari locazioni di memoria per predisporre i colori del bordo (53280) e dello sfondo (53281) dello schermo. I 16 differenti colori che possono essere usati sono numerati da 0 a 15; il ciclo FOR...NEXT che comincia alla riga 10 assegna alla variabile C proprio i valori da 1 a 15, in modo che il programma visualizzi tutti i colori eccetto il nero, che è lo 0.

I codici di tutti i colori sono mostrati nella tabella riportata in fondo a questa pagina. Dovreste ora essere in grado di capire come il comando POKE può essere usato per assegnare il colore nero allo schermo, come era stato accennato a pag. 16. Per riportare lo schermo ai suoi colori originali, dopo aver eseguito questo programma, premete contemporaneamente, il tasto RUN/STOP e il tasto RESTORE (il primo a destra della seconda fila).

Dal momento che sul Commodore sono disponibili così tante locazioni di memoria, non provocherete alcun danno caricandovi dei valori a caso; nella peggiore delle ipotesi potreste "congelare" il computer, essendo poi costretti a spegnerlo per poter riprendere il controllo; naturalmente perderete anche qualsiasi cosa si trovi nella memoria del computer in quel momento. Ci sono però molte locazioni che possono essere utilizzate efficacemente con il comando POKE: tutte le funzioni sonore del Commodore, per esempio, sono controllate in questo modo, come potrete vedere più avanti in questo libro.

Controllare il funzionamento dei tasti con POKE

Per vedere un altro esempio di funzionamento del comando POKE, provate ad immettere la prima riga della schermata seguente e, dopo aver premuto RETURN, mantenete premuto un qualsiasi tasto (non preoccupatevi del messaggio di errore):

RIPETIZIONE AUTOMATICA CON **POKE**

```
POKE 650,128
READY.

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
?SYNTAX  ERROR
READY.

POKE 650,0
READY.

A█
```

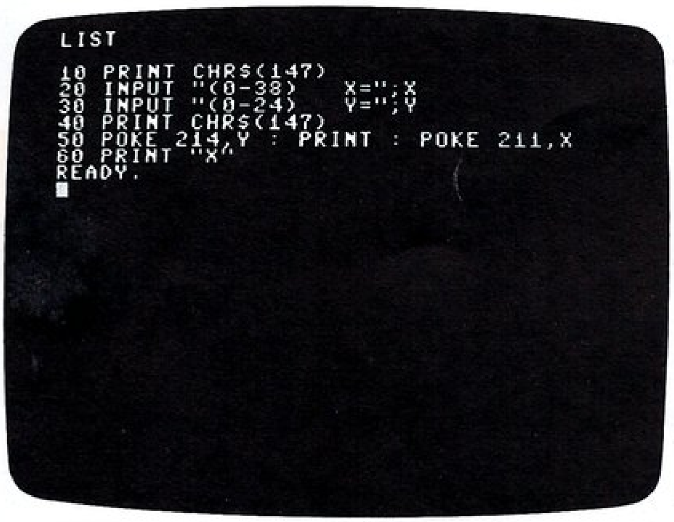
CODICE DEI COLORI PER LE ISTRUZIONI POKE

Tutti i 16 colori del Commodore possono essere prodotti con i comandi POKE terminanti con il codice del colore.

Colore	Codice POKE	Colore	Codice POKE
Nero	0	Arancione	8
Bianco	1	Marrone	9
Rosso	2	Rosa	10
Ciano	3	Grigio scuro	11
Porpora	4	Grigio	12
Verde	5	Verde chiaro	13
Blu	6	Azzurro	14
Giallo	7	Grigio chiaro	15

Come avete già potuto notare, pochi tasti ripetono automaticamente la loro funzione se li tenete premuti: si tratta della barra spaziatrice, del tasto INST/DEL e dei tasti di cursore.
La ripetizione automatica dei tasti o autorepeat, come è chiamata questa possibilità, può essere molto utile per tutti i tasti della tastiera.
La locazione di memoria 650 è quella che il Commodore usa per determinare con quali tasti deve funzionare l'autorepeat.
La schermata precedente mostra cosa succede se premete il tasto A.
Per far ritornare la tastiera nelle condizioni normali, caricate il valore 0 nella stessa locazione di memoria.
Come esempio finale dell'uso di POKE ecco un programma che accetta in ingresso un paio di coordinate (X e Y) dello schermo, muove il cursore nella posizione che avete specificato e vi stampa una "X":

PROGRAMMA DI CONTROLLO DEL CURSORE



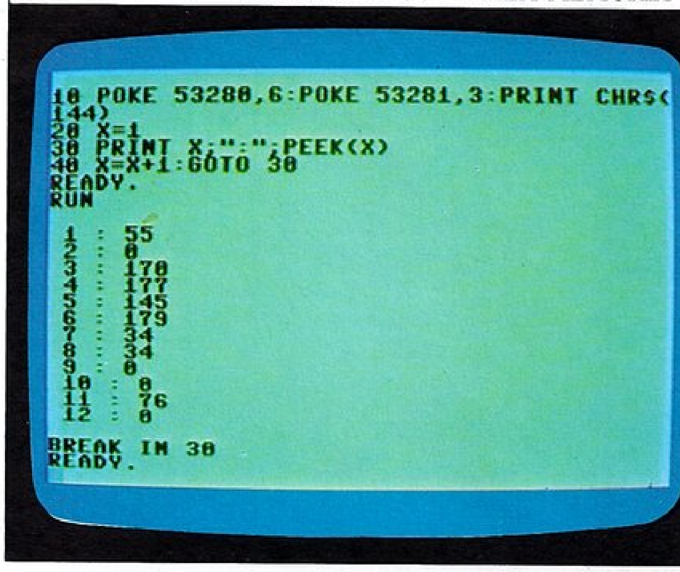
In questo caso la locazione 214 memorizza la coordinata Y del cursore e la locazione 211 la coordinata X. Questo modo di muovere il cursore sullo schermo vi tornerà molto utile nei programmi di animazione.

Curiosare nella memoria con PEEK
PEEK è un comando che produce un effetto esattamente opposto a quello di POKE. Se, per esempio, immettete:

PRINT PEEK (650)

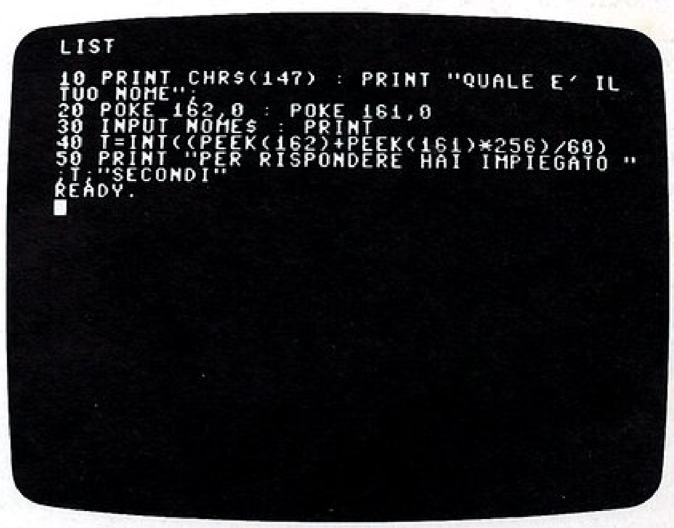
il numero visualizzato vi dirà per quali numeri è attiva la funzione di autorepeat. Questo valore sarà 128 se l'autorepeat è attivo su tutta la tastiera, o 0 se esso vale solo per i tasti di editing e la barra spaziatrice. Il comando PEEK è usato nella programmazione del Commodore per estrarre dei valori da determinate locazioni di memoria; questi valori vengono poi modificati in qualche modo e quindi memorizzati nuovamente. Potete usare un comando PEEK all'interno di un ciclo per vedere i valori memorizzati nella memoria del Commodore; questo è il compito del programma seguente: un ciclo senza fine che si snoda attraverso tutte le locazioni della memoria; questa schermata vi mostra il listato del programma e il risultato dei primi PEEK:

PROGRAMMA "VISUALIZZAZIONE DELLA MEMORIA"



Una funzione molto importante che può essere eseguita dal comando PEEK è quella di esaminare il contenuto del clock jiffy del Commodore. Non appena il computer viene acceso, il jiffy jiffy è azzerato; viene poi incrementato 60 volte al secondo, per tutto il tempo che la macchina rimane accesa.
Il conteggio del clock jiffy è memorizzato agli indirizzi 160, 161 e 162.
La locazione 162 viene modificata più velocemente, 60 volte al secondo; la locazione 161 viene incrementata ogni volta che la 162 passa dal valore 255 al valore 0; infine, la locazione 160 è quella incrementata più lentamente, ogni volta che la locazione 161 passa da 255 a 0.
Usando la locazione 162 da sola, potete fare in modo che il Commodore temporizzi degli intervalli di 4 secondi; usando le locazioni 162 e 163 il conteggio sale a 18 minuti e, con tutte e tre le locazioni si arriva a circa 3 giorni e un quarto.
Potete scoprire come utilizzare il comando PEEK con il clock jiffy se provate il programma seguente, il quale misura il tempo impiegato per rispondere ad una domanda:

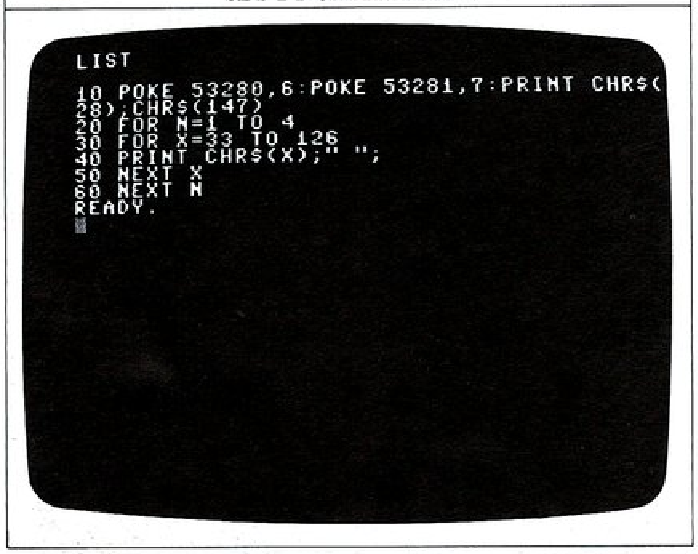
PROGRAMMA "CLOCK JIFFY"



GRAFICA DA TASTIERA

A pag. 17 avete visto come i caratteri e i controlli sulla tastiera del Commodore siano rappresentati nel computer dal codice ASCII. Potete usare questo sistema di codifica per far visualizzare al Commodore il suo "set di caratteri", cioè l'insieme di tutti i caratteri ordinati secondo il loro codice.
Ecco un programma che visualizza i caratteri con codici compresi tra 33 e 126:

SET DI CARATTERI 1





Questo programma, in realtà, ripete il set per quattro volte, riempiendo lo schermo. Potete vedere dalla tabella dei caratteri a pag. 61 che esso visualizza circa metà di tutto il set (fino al codice 126) e che omette i caratteri di controllo con codici compresi tra 0 e 32.
Il Commodore, in realtà, possiede due set di caratteri. Per visualizzare il secondo, non avete bisogno di un'altro programma: dopo aver eseguito il programma precedente, premete contemporaneamente il tasto SHIFT e il tasto C=. Dovreste ora vedere un set di caratteri in cui le lettere maiuscole sono sostituite dalle minuscole:

SET DI CARATTERI 2



Come potete vedere da queste due schermate, oltre alle lettere dalla A alla Z e ai numeri dallo 0 al 9, il Commodore ha un set molto utile di caratteri predefiniti. Potete immettere questi ultimi caratteri direttamente in un programma, utilizzando il comando PRINT, proprio come per qualsiasi altro carattere. Se osservate la tastiera, vedrete che la maggior parte dei tasti ha due di questi caratteri grafici stampati sulla parte frontale. Quello più a destra è accessibile premendo SHIFT contemporaneamente al tasto stesso. Il carattere più a sinistra può essere usato nello stesso modo, ma premendo il tasto C= invece di SHIFT. Per ottenere le lettere minuscole, dovete usare insieme i tasti SHIFT e C=, in modo da selezionare il set di carattere corrispondente.

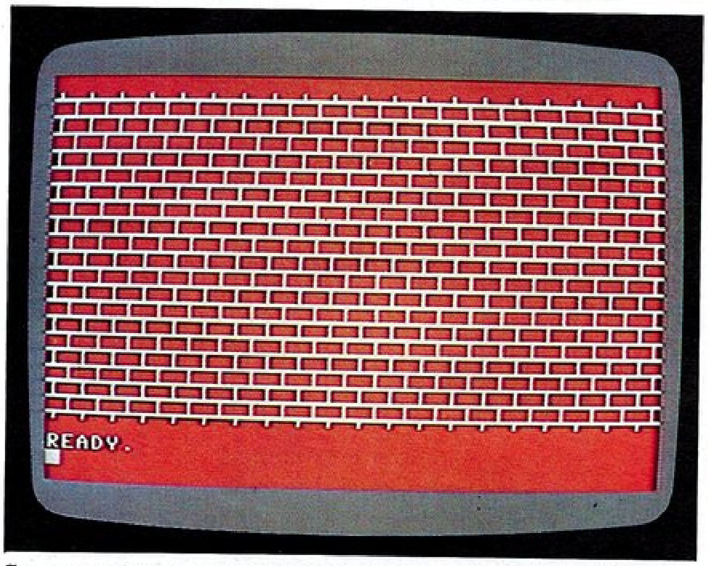
Come ottenere i simboli grafici in reverse
Se guardate la tastiera, noterete che i tasti numerici 8 e 9 sono contrassegnati con RVS ON e RVS OFF. Questi tasti vi permettono di produrre il "reverse" di ogni carattere.
Per vedere come ciò funziona, premete CTRL e 8, e quindi immettete qualche carattere sullo schermo: vedrete che i colori del testo e dello schermo sono invertiti; in questo modo voi potete visualizzare un carattere che ha i colori invertiti rispetto a quelli prodotti di solito. Premendo CTRL e 9 si disattiva questa funzione. Se usate CTRL e 8 all'interno di una stringa, essa vi apparirà come un simbolo di controllo rappresentato da una R in reverse. RVS ON e RVS OFF possono essere attivati anche con PRINT CHR$. I caratteri grafici della tastiera sono disposti in gruppi con linee di spessore variabile.
Se non potete vedere il particolare carattere che vi serve per completare un disegno, può darsi che ciò sia dovuto al fatto che vi occorre un carattere che è il reverse di un altro. Un carattere in reverse che è particolarmente utile è lo spazio: esso produce un quadrato pieno che può essere usato per costruire aree colorate sullo schermo.

Come disegnare con la grafica della tastiera
Potete usare i caratteri grafici del Commodore, in un programma, per costruire immagini sullo schermo.
Il seguente programma usa solo due di questi caratteri, uno ha la forma di una piccola T maiuscola (riga 50) e l'altro quella di una T maiuscola rovesciata (riga 49).
Questi due caratteri sono riportati sui tasti E ed R:

PROGRAMMA "GRAFICA DELLA TASTIERA"

```
LIST

10 PRINT CHR$(147) : POKE 53280,12:POKE
53281,2
20 C=1 : PRINT CHR$(5);
30 FOR K=1 TO 20 : FOR D=1 TO 40
40 IF C=1 THEN PRINT"┴"; : GOTO 60
50 PRINT "┬";
60 C=1-C
70 NEXT D
80 C=1-C
90 NEXT K
READY.
```



Se eseguite il programma, esso produrrà il disegno piuttosto accurato di un muro.
Se selezionate ancora il set di caratteri minuscoli, il programma funzionerà sempre perfettamente ma, in questo caso, dovete ricordare di scrivere tutte le parole chiave del BASIC in minuscolo (senza schiacciare il tasto SHIFT), altrimenti il programma non funzionerà assolutamente.
Potete usare i caratteri grafici della tastiera anche per costruire altre figure sullo schermo, per mezzo di cicli.
Il prossimo programma, per esempio, produce una pila triangolare di linee stampando una serie di caratteri grafici.
Il numero di caratteri che sono stampati su ogni riga è determinato dall'intervallo di valori che assume la variabile N:

PROGRAMMA TRIANGOLO

```
LIST

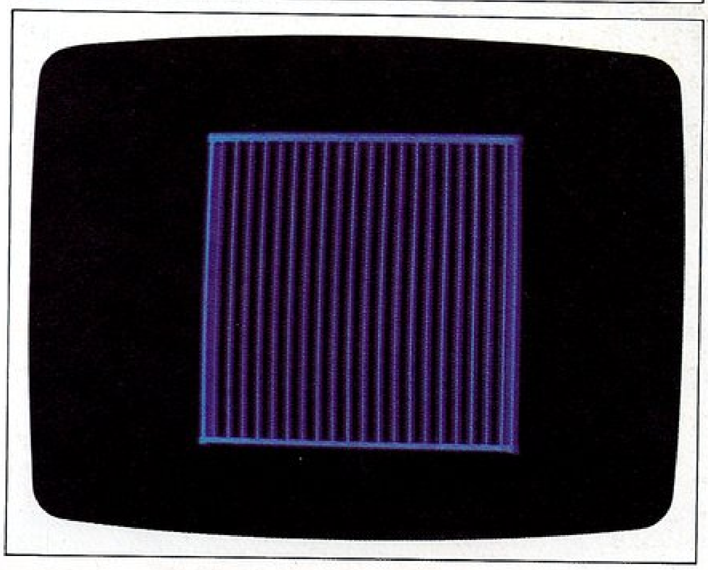
10 PRINT CHR$(147);CHR$(5)
20 POKE 53280,5:POKE 53281,0
30 FOR X=1 TO 20
40 FOR N=1 TO 2*X
50 PRINT "▬";
60 NEXT N
70 PRINT
80 NEXT X
READY.
```

Coi cicli potete produrre molte figure senza dover scrivere nel programma l'intera figura da visualizzare. Il prossimo programma disegna e riempie un rettangolo:

PROGRAMMA RETTANGOLO

```
LIST

10 PRINT CHR$(147);CHR$(31)
20 POKE 53280,0:POKE 53281,0
30 PRINT "▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁"
40 FOR Y=1 TO 18
50 PRINT "|||||||||||||||||||||"
60 NEXT Y
70 PRINT "▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔"
READY.
```

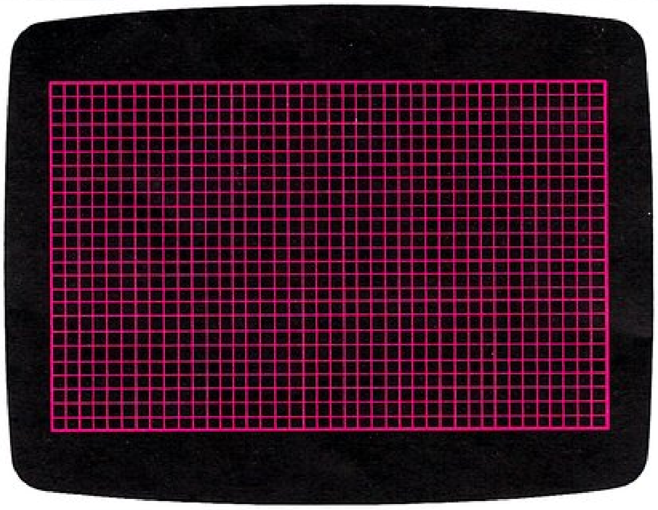


LA MEMORIA VIDEO

Ora che ne sapete di più sui set di caratteri del Commodore, potete passare ad un metodo più versatile di PRINT CHR$ per visualizzare i caratteri sullo schermo.
Questa nuova tecnica usa POKE per definire sia il carattere che il colore da visualizzare in una posizione dello schermo.

Memoria video e memoria colore
Lo schermo del Commodore può essere pensato come suddiviso in una griglia su cui sono visualizzati i caratteri.
Nel computer ci sono due aree di memoria, le cui locazioni corrispondono ai punti della griglia: la memoria video e la memoria colore.
La prima contiene tutti i caratteri che devono essere visualizzati e la seconda il loro colore. Sia i caratteri che i colori possono essere selezionati per mezzo di comandi POKE. Poiché lo schermo è suddiviso in 25 linee di 40 caratteri, la memoria video e la memoria colore occupano entrambe 1000 byte.
La memoria video inizia alla locazione 1024 e forma un blocco continuo fino alla locazione 2023.
In modo analogo, la memoria colore, va dalla locazione 55296 fino a 56295.
La griglia seguente mostra le posizioni possibili sullo schermo (la griglia a pag. 60 dà anche i valori da memorizzare con POKE):

GRIGLIA DI TESTO PER IL COMMODORE



Per visualizzare un carattere sullo schermo con un programma dovete immettere una riga come questa:

```
90 POKE 1464,81 : POKE 55696, 1
```

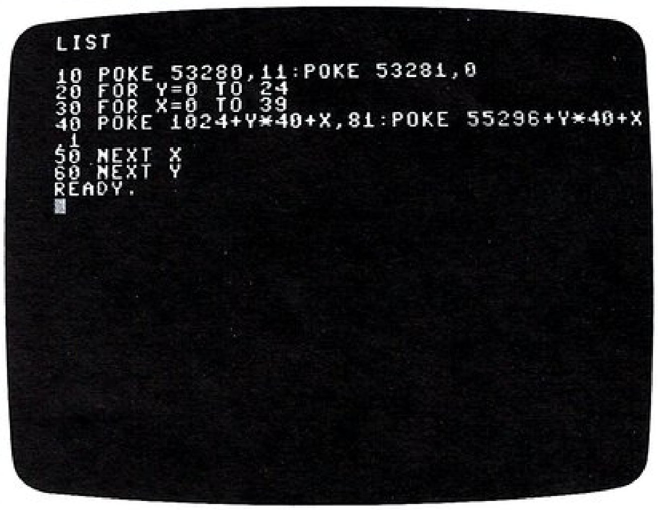
Essa inserisce un carattere alla riga 10 dello schermo nella posizione 0, cioè al margine sinistro. Se cercate questo codice (81) nella tabella di pag. 60, scoprirete che corrisponde a un circoletto pieno. Il secondo POKE fissa il colore; senza di esso il simbolo sarebbe invisibile poiché risulterebbe visualizzato con lo stesso colore dello sfondo. Potete vedere sulla tabella dei colori a pag. 32 che il valore 1, il quale appare alla fine di questa riga di programma, corrisponde al bianco.

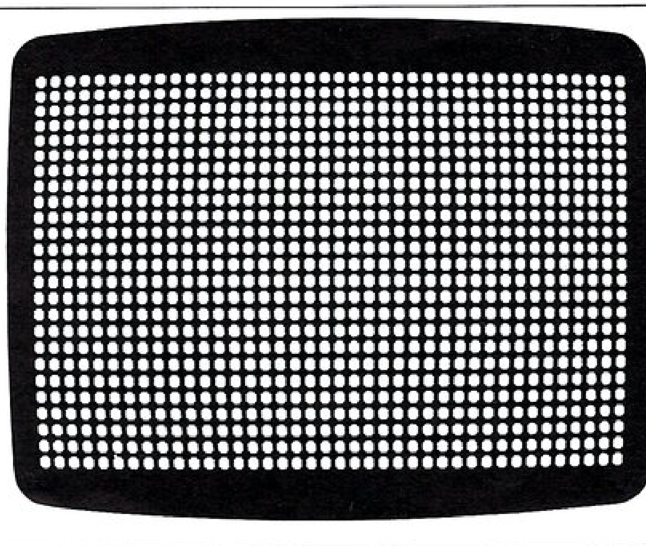
Un modo per scrivere nelle locazioni di memoria
Se doveste ricordarvi tutti i valori relativi ai comandi di POKE ogni volta che li usate in un programma, impieghereste senz'altro molto tempo. Tuttavia, conoscendo la posizione delle prime locazioni di queste due zone di memoria, potete ottenere rapidamente il valore da usare con POKE per ogni altro punto.
Se Y è il numero di linea dello schermo e X è la posizione nella linea, la riga seguente visualizzerà una pallina bianca alla posizione di coordinate X, Y:

```
40 POKE 1024+Y*40+X,81 : POKE 55296+Y*40+X,1
```

Essa accetta qualsiasi valore di X (coordinata orizzontale) e di Y (coordinata verticale) per visualizzare un carattere, sempre che questi rientrino nei limiti della mappa video. Variando ripetutamente sia X che Y, potete visualizzare un carattere in una posizione qualsiasi dello schermo:

PROGRAMMA CHE VISUALIZZA DEI CARATTERI



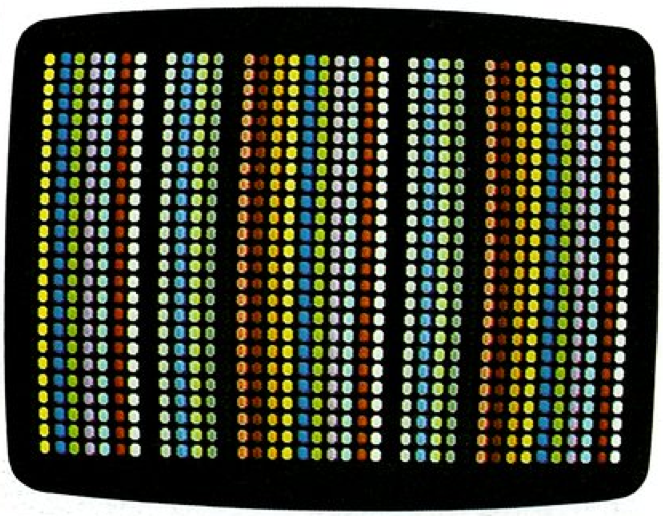


Potete provare ad aggiungere i colori. Potreste voler semplicemente passare dal bianco ad un altro colore; provate invece a modificare il carattere finale della riga 40 in questo modo:

```
40 POKE 1024+Y*40+X,81:POKE 55296+Y*40+X,X
```

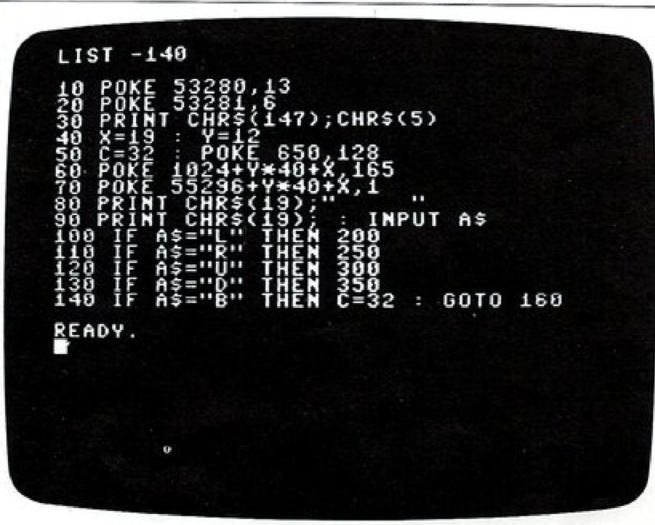
Ora il colore dipende dalla posizione e varia per ogni carattere. Quando X supera 15, il computer fa ripartire la sequenza dei calcoli dall'inizio.
Otterrete la stessa visualizzazione dello schermo precedente, ma questa volta con tutti i colori del Commodore:

VISUALIZZAZIONE DI CARATTERI COLORATI

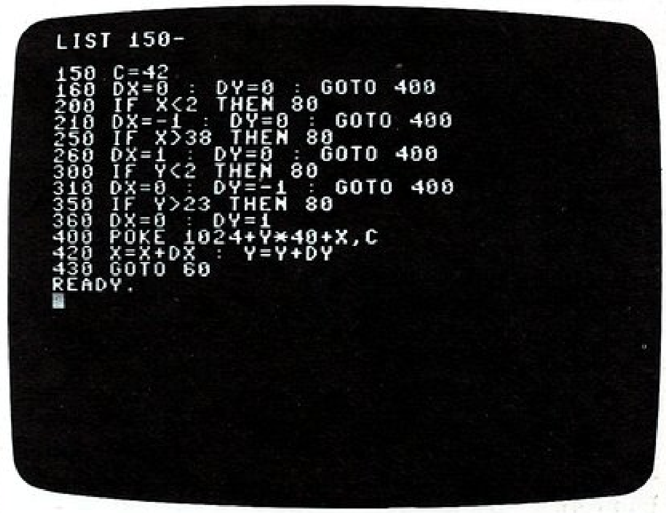


Come usare POKE per la grafica
Usando POKE per visualizzare un carattere e un colore sullo schermo, potete costruire le vostre immagini grafiche. Il seguente programma vi permette di disegnare sullo schermo: esso è un semplice programma esemplificativo, che usa uno dei caratteri del testo (un asterisco), per tracciare delle righe. Il programma usa POKE per cambiare la posizione del cursore ogni volta che immette un'istruzione:

PROGRAMMA GRAFICO

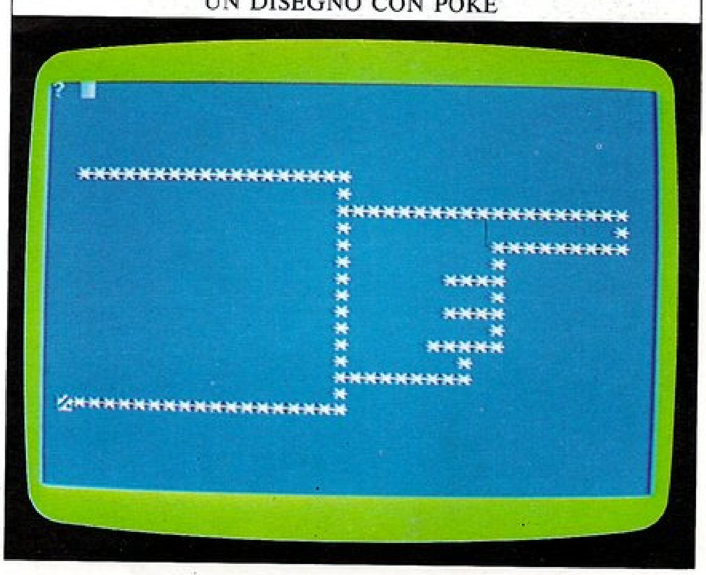


PROGRAMMA GRAFICO



Potete controllare il movimento del cursore con U, D, L e R (Up, Down, Left e Right). Quando avete scelto la direzione in cui volete muovere il cursore, premete RETURN.
Per fare un altro passo nella stessa direzione dovete solo premere RETURN un'altra volta. Se volete muovere il cursore di alcuni passi in una direzione qualsiasi, potete usare la funzione di autorepeat: ciò significa che dovete solo mantenere premuto il tasto RETURN ed il cursore si muoverà rapidamente nell'ultima direzione che avrete scelto.
Per disegnare una linea mentre il cursore si muove, immettete un asterisco, premete RETURN e quindi muovete il cursore nel solito modo; per cancellare delle linee, se avete fatto degli errori, immettete B, premete sempre RETURN e il cursore visualizzerà degli spazi nelle posizioni che raggiunge. La visualizzazione è limitata a 40x25 posizioni di carattere, tuttavia, benché il disegno sia rappresentato in bassa risoluzione, il programma è molto veloce.
Ecco un esempio del tipo di visualizzazione che potete ottenere:

UN DISEGNO CON POKE

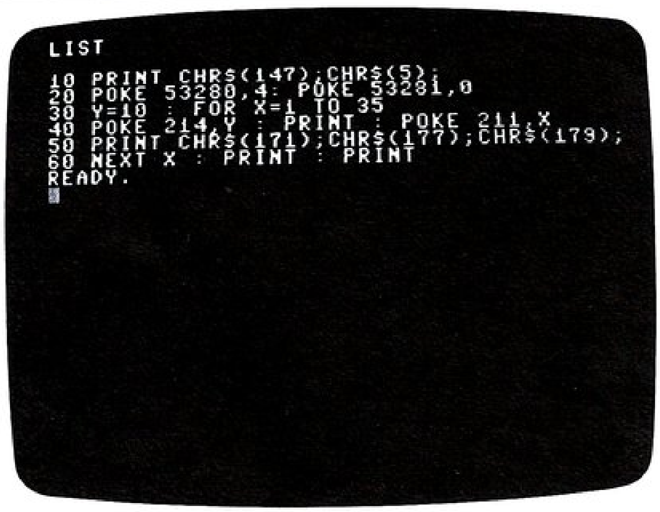


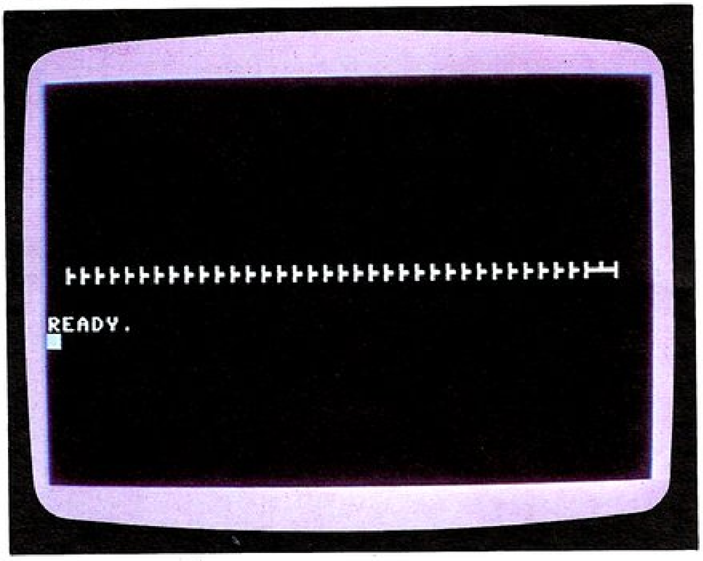
ANIMAZIONE

L'animazione consiste nel programmare il computer per posizionare un carattere sullo schermo e quindi muoverlo con una serie di piccoli spostamenti. Ci sono due modi per fare ciò, usando sia POKE con il comando PRINT che POKE da solo per selezionare la posizione, il carattere ed il colore. Entrambi questi metodi usano dei cicli per modificare la posizione del carattere nel modo che voi avete specificato.

Ecco un semplice programma di animazione che vi mostra alcuni dei problemi che dovrete superare per controllare il movimento:

ANIMARE UN CARATTERE

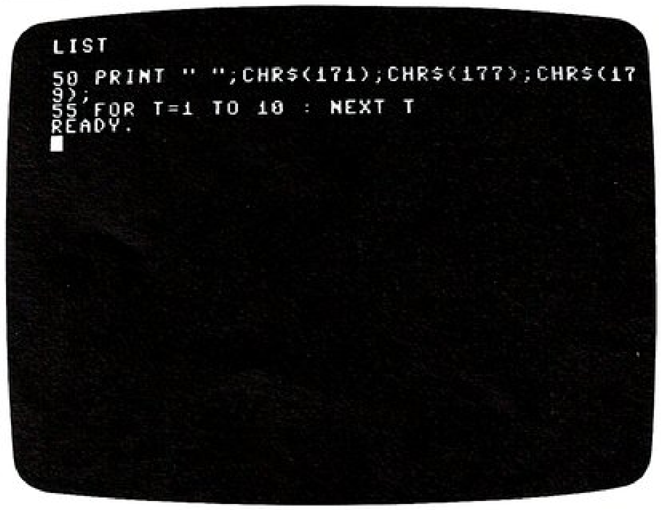




Come cancellare le immagini precedenti e controllare la velocità

Quando eseguite il programma precedente esso non fa esattamente quello che voi volete. Esso sposta sì il carattere da destra verso sinistra, cambiando la posizione del cursore per mezzo di POKE, ma non cancella le vecchie immagini, dal momento che non avete detto al computer di farlo. Ciò significa che, dopo ogni movimento, una parte dell'immagine precedente rimane sullo schermo. In alcuni programmi potreste voler mantenere queste vecchie immagini per ottenere un particolare effetto, ma nella maggior parte dei casi, preferirete eliminarle. Un'altra cosa che avrete notato, è la velocità con cui l'oggetto si muove lungo lo schermo: è troppo alta per la maggioranza degli scopi. Tutte e due questi problemi possono essere eliminati inserendo le seguenti righe nel programma precedente, che è ancora in memoria:

MODIFICHE PER MIGLIORARE L'IMMAGINE



Lo spazio presente alla riga 50 cancella la parte sinistra dell'immagine, mentre un ciclo di ritardo rallenta il movimento.

Il movimento verticale può essere ottenuto nello stesso modo; questa volta, però, la coordinata X è mantenuta costante e quella Y viene modificata da un ciclo. Potete far muovere un oggetto verso l'alto immettendo una riga di questo tipo:

```
50 POKE 214,24-Y:POKE 211,X
```

Mentre Y aumenta, l'oggetto è visualizzato sempre più in alto sullo schermo; per cancellare le vecchie immagini deve essere visualizzato uno spazio, muovendo il cursore indietro di una posizione ad ogni movimento.

Movimenti in due direzioni

Fino ad ora avete realizzato delle animazioni che utilizzavano FOR...NEXT per modificare la coordinata X o la coordinata Y di un oggetto, spostandolo solo in una direzione. Questa idea è ottima nel suo piccolo, ma diventa inutile quando, per esempio, volete far rimbalzare una pallina da un lato all'altro dello schermo. Il più semplice, e probabilmente anche uno dei migliori, metodi per realizzare questo movimento in due direzioni consiste nell'utilizzare un'ulteriore variabile in aggiunta alle coordinate X e Y; questa nuova variabile memorizza la direzione del movimento dell'oggetto. Il programma seguente usa questa tecnica con la variabile DX per muovere una pallina da un lato dello schermo all'altro:

PROGRAMMA "PALLINA CHE RIMBALZA"

```
LIST

10 PRINT CHR$(147);CHR$(5)
20 Y=10 : X=20 : DX=1 : POKE 53280,6 : P
OKE 53281,12
30 POKE 214,Y : PRINT : POKE 211,X
40 PRINT " ";CHR$(113);" ";
50 X=X+DX
60 FOR T=1 TO 10 : NEXT T
70 IF X<1 THEN DX=-DX
80 IF X>36 THEN DX=-DX
90 GOTO 30
READY.
```

In questo programma, la riga 20 memorizza in X e Y le coordinate iniziali e in DX la direzione da sinistra a destra. La riga 30 posiziona il cursore nella posizione corrispondente al valore aggiornato di X e Y; la riga 40 visualizza la pallina e cancella anche l'immagine precedente alla sinistra o alla destra dell'attuale, visualizzando uno spazio in quel punto. La riga 50, infine, aggiorna la coordinata X sommandole il valore di DX.
Dal momento che DX attualmente contiene il valore 1, X viene incrementata di un'unità e la pallina si sposta di una posizione verso destra; se D valesse -1, allora la stessa riga muoverebbe la pallina di una posizione a sinistra. Le righe 70 e 80 invertono la direzione della pallina, se questa si trova al bordo dello schermo, cambiando il segno di DX da +1 a -1 e viceversa.

Animazione con POKE

Così come è possibile usare PRINT CHR$ per realizzare un'animazione, potete anche caricare (POKE) dei caratteri direttamente dal set di caratteri grafici allo schermo, cancellandoli ogni volta per simulare il movimento:

PROGRAMMA "PISTA COLORATA"

```
LIST -140

10 PRINT CHR$(147)
20 POKE 53280,6 : POKE 53281,6
30 FOR X=10 TO 30
40 POKE 1024+6*40+X,81 : POKE 55296+6*40
+X,6
50 POKE 1024+15*40+X,81 : POKE 55296+15*
40+X,6
60 NEXT X
70 FOR Y=7 TO 14
80 POKE 1024+Y*40+10,81 : POKE 55296+Y*4
0+10,6
90 POKE 1024+Y*40+30,81 : POKE 55296+Y*4
0+30,6 : NEXT Y
100 FOR X=10 TO 30
110 POKE 55296+6*40+X,1
120 FOR T=1 TO 5 : NEXT T
130 POKE 55296+6*40+X,6
140 NEXT X

READY.
```

PROGRAMMA "PISTA COLORATA"

```
LIST 150-

150 FOR Y=7 TO 14
160 POKE 55296+Y*40+30,1
170 FOR T=1 TO 5 : NEXT T
180 POKE 55296+Y*40+30,6
190 NEXT Y
200 FOR X=10 TO 30
210 POKE 55296+15*40+(40-X),1
220 FOR T=1 TO 5 : NEXT T
230 POKE 55296+15*40+(40-X),6
240 NEXT X
250 FOR Y=7 TO 14
260 POKE 55296+(21-Y)*40+10,1
270 FOR T=1 TO 5 : NEXT T
280 POKE 55296+(21-Y)*40+10,6
290 NEXT Y
300 GOTO 100
READY.
```

In questo programma una pallina si muove su una pista rettangolare; il listato appare ermetico, ma, quando sono richiesti dei movimenti complessi, l'uso di POKE risulta molto più semplice rispetto al comando PRINT CHR$, anche se questo significa dover calcolare le locazioni delle memorie video e colore. È, inoltre, più semplice sviluppare i programmi modificando dei valori per il comando POKE.
Le righe 10 e 20 cancellano lo schermo e selezionano il colore blu per lo sfondo. Le righe dalla 30 alla 90 visualizzano una cornice di palline blu sullo schermo; le righe da 100 a 290, quindi, cambiano il colore di ogni pallina, a turno, in senso orario, da blu (il colore dello sfondo) a bianco e, dopo un breve ciclo di ritardo, ancora da bianco a blu. Se immettete le righe seguenti, per cambiare il colore dello schermo, l'intero processo diventerà visibile:

```
20 POKE 53280,6:POKE 53281,0
```

Dopo aver effettuato la modifica, l'immagine assomiglierà a questa, con la pallina bianca che viaggia sulla pista blu:

VISUALIZZAZIONE DELLA TRACCIA COLORATA



MEMORIZZARE I DATI 1

I dati necessari a un programma possono essere acquisiti mentre il programma è in esecuzione con INPUT oppure essere immagazzinati nel programma stesso. I comandi usati per memorizzare i dati sono semplici: il dato è contenuto nell'istruzione DATA e letto da READ. Questo programma vi mostra la tecnica:

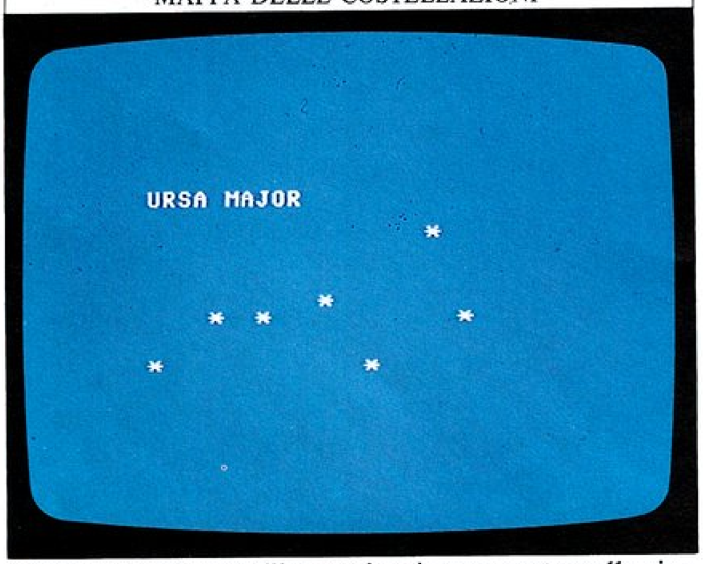
PROGRAMMA "COSTELLAZIONI"

```
LIST

10 POKE 53280,6:POKE 53281,6:PRINT CHR$(
147);CHR$(5)
20 POKE 214,6 : PRINT : POKE 211,7
30 PRINT "URSA MAJOR"
40 DATA 8,25,12,18,13,11,13,14,13,27,16,
7,16,21
50 NUMERO=7
60 FOR N=1 TO NUMERO
70 READ Y,X
80 POKE 214,Y : PRINT : POKE 211,X
90 PRINT "*"
100 FOR T=1 TO 500 : NEXT T
110 NEXT N
120 PRINT CHR$(31)
READY.
```

Quando eseguirete il programma vedrete sul vostro schermo la mappa di un gruppo di stelle generata dal computer; si tratta della costellazione dell'Orsa Maggiore, conosciuta anche come il Grande Carro:

MAPPA DELLE COSTELLAZIONI



L'informazione per l'immagine è contenuta nella riga 40 sotto forma di 14 coordinate. La riga 70 dice al computer di leggere (con READ) i dati della riga 40, e di considerare il gruppo di dati come coppie di valori che il programma indicherà con X e Y. La riga 50, inoltre, indica al computer che esistono sette di queste coppie. Con un programma come questo è facile introdurre nuovi dati per far visualizzare al computer una nuova mappa. Ecco le modifiche da fare per ottenere due nuove mappe:

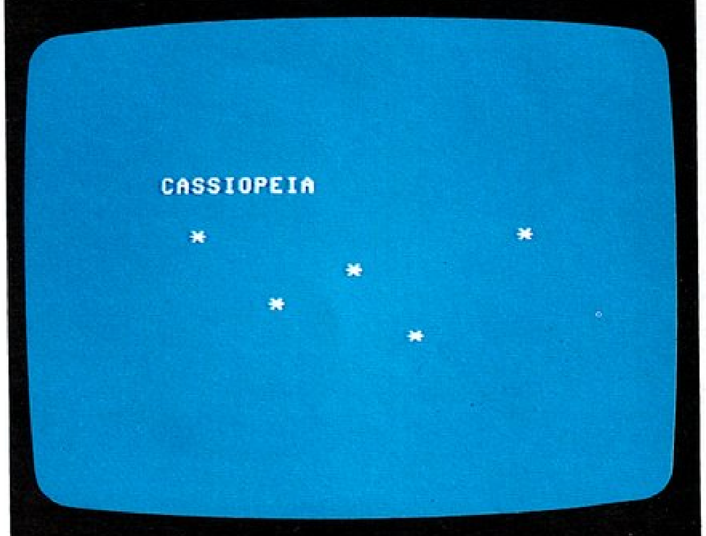
NUOVE RIGHE DEL PROGRAMMA "COSTELLAZIONE"

```
LIST 30-50

30 PRINT "CASSIOPEIA"
40 DATA 9,9,9,30,11,19,13,14,15,23
50 NUMERO=5

READY.


30 PRINT "LYRA"
40 DATA 8,23,12,20,14,16,18,20,19,16
50 NUMERO=5
```





Quando usate istruzioni DATA è importante dire al computer quanti dati esso deve leggere. La riga 60 nel programma delle costellazioni mostra come fare ciò: essa fissa il limite per il numero di coppie di coordinate che sono memorizzate nella riga DATA cosicché, quando il computer ha visualizzato l'ultima stella, esso pone fine al programma. Senza questa precauzione il computer cercherebbe di leggere dei dati inesistenti e il programma si interromperebbe con un messaggio di errore:

```
? OUT OF DATA ERROR IN 70
```

Come memorizzare numeri e parole assieme

Anche le parole possono essere memorizzate e lette usando righe di DATA, cosicché potete memorizzare assieme numeri e parole: il nome degli amici ed il loro numero di telefono o la data del compleanno, per esempio.

Questo potrebbe far nascere dei problemi, poiché devono essere utilizzati due differenti tipi di dati, numeri e stringhe. Una attenta organizzazione delle istruzioni DATA e READ, però, può aggirare l'ostacolo:

PROGRAMMA "AGENDA TELEFONICA"

```
10 PRINT CHR$(147)
20 DATA A.ROSSI,167,G.ANTONELLI,244,R.CA
STELLI,144
30 DATA S.BERNA,119,M.ROSSINI,223,T.ARMA
TI,240,P.CATTANEO,115
40 DATA D.PRATESI,444
50 PRINT "AGENDA TELEFONICA PERSONALE"
60 FOR T=1 TO 1000 : NEXT
70 PRINT CHR$(147) : POKE 214,4 : PRINT
80 PRINT "LISTA COMPLETA ...PREMERE 1":
PRINT
90 PRINT "LISTA PARZIALE ...PREMERE 2"
100 PRINT : INPUT "SCELTA";SCELTA
110 IF SCELTA=2 THEN 190
120 PRINT CHR$(147)
130 PRINT : PRINT : PRINT
140 FOR C=1 TO 8
150 READ NOMES$,N
160 PRINT TAB(6);NOMES$;TAB(20);N :PRINT
170 NEXT C
180 END

READY.
```

```
LIST 190-

190 PRINT CHR$(147) :POKE 214,4: PRINT :
PRINT "SCRIVI L' INIZIALE E IL COGNOME"
200 INPUT E$
210 B$="NOME NON TROVATO"
220 FOR C=1 TO 8
230 READ NOMES$,N$
240 IF NOMES$=E$ THEN B$=NOMES$+"......."+
N$
250 NEXT C : PRINT CHR$(147)
260 PRINT B$ : FOR T=1 TO 2000
270 NEXT : RESTORE : PRINT CHR$(147) : G
OTO 50
280 NEXT : RESTORE : PRINT CHR$(147) : G
OTO 50
READY.
```

Questo programma memorizza una agenda personale di numeri di telefono. Nomi e numeri telefonici sono contenuti nelle righe da 20 a 40. Le righe da 50 a 90 visualizzano il titolo del programma e poi offrono una scelta tra diverse funzioni. Se immettete:

```
1 RETURN
```

il computer visualizza l'intera agenda telefonica:

VISUALIZZAZIONE DELL'AGENDA TELEFONICA

```
A.ROSSI         167
G.ANTONELLI     244
R.CASTELLI      144
S.BERNA         119
M.ROSSINI       223
T.ARMATI        240
P.CATTANEO      115
D.PRATESI       444

READY.
```

Se immettete:

```
2 RETURN
```

il programma esegue le righe da 190 a 270 e vi chiede di inserire l'iniziale del nome e il cognome.

Se il computer trova che il nome (ENTRY$) che avete immesso, è uguale ad uno dei nomi (NAME$) contenuti nelle istruzioni DATA, assegnerà ad una nuova stringa (B$) il valore di NAME$ più una linea di punti e un numero telefonico. Se il programma non trova ENTRY$, B$ rimane invariato e vale "Nome non trovato" (valore assegnatole alla riga 210) e questa stringa è visualizzata al termine. Siccome volete aggiungere al nome una linea di punti ed il numero telefonico, utilizzando una sola istruzione (riga 240), quest'ultimo deve essere contenuto in una variabile stringa (N$) invece che nella variabile numerica (N) usata alla riga 150. Provate a sostituire N con N$ alle righe 230 e 240, vedrete apparire un messaggio di errore:

```
? TYPE MISMATCH ERROR IN 240
```

poiché non si possono sommare stringhe e numeri in una stessa istruzione. Potete estendere il programma fino a contenere una lista molto più lunga di nomi e numeri telefonici, inserendoli nelle righe DATA, e quindi modificando i limiti dei due cicli alle righe 140 e 220. Per visualizzare la lista completa, se essa è più lunga di una schermata, premete CTRL non appena compaiono le prime righe, così da rallentarne lo scorrimento. La riga 220 usa un comando che incontrate per la prima volta: RESTORE. Senza di esso, il programma viene eseguito correttamente una sola volta, dopo di che produrrà un messaggio di errore. Potete trovare la spiegazione di questo fatto a pag. 43.

MEMORIZZARE I DATI 2

Quando i dati che avete immagazzinato nelle istruzioni DATA si riferiscono a caratteri e simboli grafici, potete costruire, usando READ, dei disegni anche molto complessi. In queste due pagine imparerete come memorizzare questi dati per produrre più facilmente immagini immobili o animate. Il programma seguente usa DATA per memorizzare la pianta di un labirinto; esso mostra anche un nuovo modo di usare il comando GOTO:

PROGRAMMA "LABIRINTO"

```
LIST -170

10 POKE 53280,8:POKE 53281,0:PRINT CHR$(
147):CHR$(5):PRINT:PRINT
20 FOR Y=1 TO 13:FOR X=1 TO 13
30 READ M : PRINT TAB(12);
40 ON M GOTO 60,70,80,90,100,110
50 ON M-6 GOTO 120,130,140,150,160
60 PRINT CHR$(98); : GOTO 170
70 PRINT CHR$(99); : GOTO 170
80 PRINT CHR$(173); : GOTO 170
90 PRINT CHR$(176); : GOTO 170
100 PRINT CHR$(174); : GOTO 170
110 PRINT CHR$(189); : GOTO 170
120 PRINT CHR$(177); : GOTO 170
130 PRINT CHR$(171); : GOTO 170
140 PRINT CHR$(178); : GOTO 170
150 PRINT CHR$(179); : GOTO 170
160 PRINT " ";
170 NEXT X

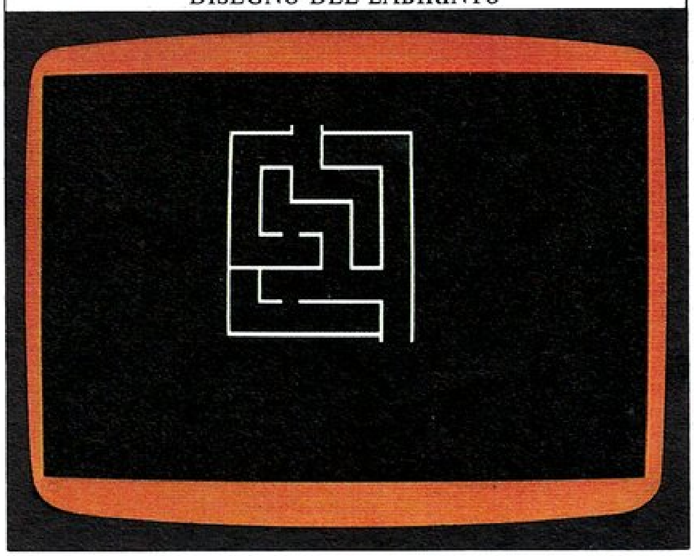
READY.
```

```
LIST 180-

180 PRINT
190 NEXT Y
200 DATA 4,2,2,2,6,11,8,2,2,2,2,2,5
210 DATA 1,11,11,11,11,11,1,11,11,11,11,
11,1
220 DATA 1,11,4,2,5,11,3,2,2,2,5,11,1
230 DATA 1,11,1,11,1,11,11,11,11,11,1,11
1
240 DATA 1,11,1,11,3,2,2,2,5,11,1,11,1
250 DATA 1,11,1,11,11,11,11,11,1,11,1,11
1
260 DATA 1,11,3,2,11,2,5,11,1,11,1,11,1
270 DATA 1,11,11,11,11,11,1,11,1,11,1,11
1
280 DATA 8,2,9,2,2,2,6,11,3,2,6,11,1
290 DATA 1,11,1,11,11,11,11,11,11,11,11,
11,1
300 DATA 1,11,3,2,11,2,2,2,2,2,5,11,1
310 DATA 1,11,11,11,11,11,11,11,11,11,1,
11,1
320 DATA 3,2,2,2,2,2,2,2,2,2,10,11,1
READY.
```

Per disegnare il labirinto, il programma utilizza alcuni dei caratteri grafici presenti nella memoria del Commodore. Invece di memorizzare nelle istruzioni DATA i codici ASCII dei caratteri utilizzati, il programma assegna a questi ultimi un proprio codice, con valori compresi tra 1 e 11. Quando devono essere memorizzati molti dati può rivelarsi molto importante limitare al massimo il numero delle istruzioni DATA; l'unico problema che questa tecnica comporta è che i dati devono essere decodificati prima che possano essere usati: nel nostro caso, per esempio, il programma deve dire al computer quale carattere visualizzare in corrispondenza ad ogni nuovo codice. Una tale operazione è effettuata alle righe 40 e 50, per mezzo dell'istruzione ON...GOTO. La variabile M alla riga 40 contiene il dato codificato che è stato appena letto (READ) dalla riga 30. La riga 40 implica che, se il valore di M è uno, il programma salti al primo numero di riga specificato nella lista che segue la parola chiave GOTO; se il valore di M è 2, invece, il programma salterà alla seconda riga della lista, e così via. Se M dovesse assumere un valore maggiore del numero di elementi contenuti in quella lista, la riga 40 verrebbe ignorata, e il computer passerebbe ad eseguire la riga 50. Quest'ultima sottrae 6 dal valore di M e seleziona una riga da una nuova lista di numeri di riga.
Ognuna delle righe specificate nelle istruzioni ON...-GOTO visualizza l'appropriato carattere grafico, e permette al programma di continuare a leggere i successivi caratteri, necessari per disegnare il labirinto:

DISEGNO DEL LABIRINTO



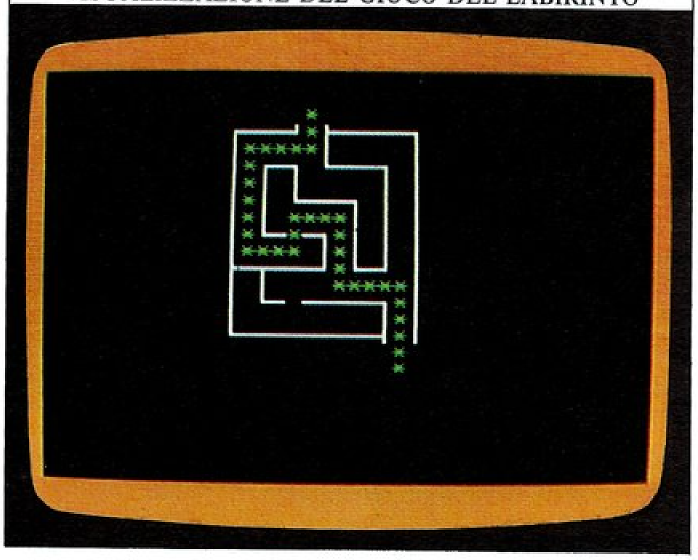
Una volta che siete in grado di ottenere un disegno come questo, potete usarlo in uno qualsiasi dei vostri programmi. Per mezzo dei caratteri grafici potete realizzare disegni anche molto complessi, tabelle e grafici (ricordatevi di inserire i comandi PRINT in un ciclo per risparmiare memoria). I tasti di cursore del Commodore vi permetteranno di muovere il cursore attraverso il labirinto per individuarne l'uscita: utilizzando un certo numero di labirinti differenti e una routine di temporizzazione, potete sviluppare questo sistema di memorizzazione dei dati per inventare un semplice gioco.

CARATTERI GRAFICI UTILIZZATI PER IL LABIRINTO

Nel programma del labirinto, l'immagine è costruita utilizzando i codici di alcuni caratteri grafici.

Carattere	Codice	Carattere	Codice
	1		7
	2		8
	3		9
	4		10
	5		
	6		

VISUALIZZAZIONE DEL GIOCO DEL LABIRINTO



Utilizzare l'istruzione DATA per l'animazione

Un'altra applicazione grafica in cui le istruzioni DATA possono rivelarsi molto utili, è la memorizzazione dei dati relativi alle figure da animare.
Per poter rileggere continuamente gli stessi dati dovreste utilizzare anche il comando RESTORE.
Ecco un programma che usa questa tecnica:

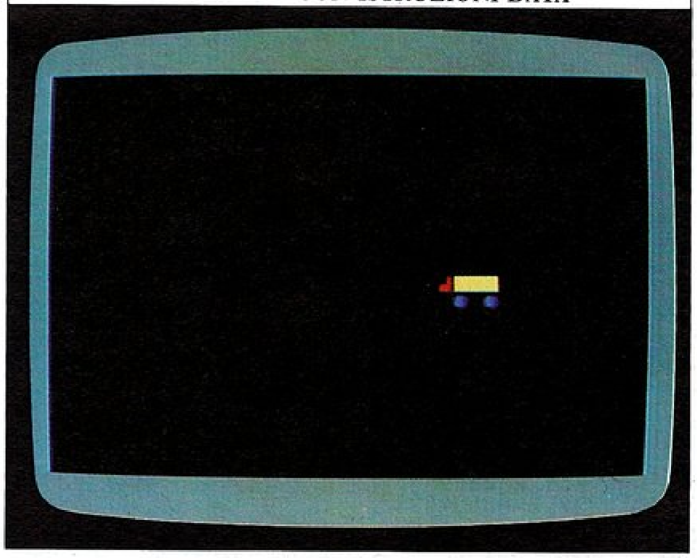
PROGRAMMA DI ANIMAZIONE CON DATA

```
LIST

10 PRINT CHR$(147)
20 POKE 53280,12:POKE 53281,0
30 X=10 : Y=7 : DX=1 : DY=-1
40 IF X<3 THEN DX=-DX
50 IF X>35 THEN DX=-DX
60 IF Y<2 THEN DY=-DY
70 IF Y>19 THEN DY=-DY
80 X=X+DX : Y=Y+DY : RESTORE
90 POKE 214,Y : PRINT : POKE 211,X
100 READ A$ : PRINT A$
110 POKE 214,Y+1 : PRINT : POKE 211,X+1
120 READ B$ : PRINT B$
130 FOR T=1 TO 10 : NEXT T
140 PRINT CHR$(147) : GOTO 40
150 DATA "[illegible]","[illegible]● ●"
READY.
```

Questo programma fa uso di una tecnica di animazione simile a quella mostrata a pag. 39 ma, in questo caso, il programma è strutturato in modo differente. La riga 10 cancella lo schermo e la 20 predispone i colori; la riga 30 inizializza i valori delle coordinate X e Y e delle direzioni del movimento. Le righe dalla 40 alla 70 controllano che le coordinate siano all'interno dei valori possibili (le posizioni sullo schermo); la riga 80 aggiorna X e Y sommandovi i valori di DX e DY, quindi reinizializza (RESTORE) il puntatore alla prima istruzione DATA da leggere (capirete il significato di questa operazione tra un attimo). Le righe 100 e 120 leggono i dati relativi alle figure, prelevandoli dall'istruzione DATA alla riga 150, quindi visualizzano e colorano l'immagine sullo schermo. La riga 130 esegue un breve ciclo di ritardo prima che la riga 140 cancelli lo schermo.

ANIMAZIONE CON ISTRUZIONI DATA



Come reinizializzare il puntatore ai dati

Potete inserire delle istruzioni DATA in un punto qualsiasi del vostro programma: il Commodore le considererà come se fossero tutte di seguito. Ogni volta che il calcolatore esegue un comando READ, esso legge il dato successivo in una istruzione DATA. Cosa succede però quando il computer arriva alla fine delle istruzioni DATA?
In un programma che utilizza una sola volta i dati memorizzati non c'è nessun problema, ma se volete che il computer utilizzi un certo numero di volte gli stessi dati, dovete usare il comando RESTORE per fare in modo che il puntatore ai dati punti nuovamente alla prima delle istruzioni DATA. Ecco perché era necessario utilizzare questo comando nel programma per l'agenda telefonica riportato a pag. 41, e nei precedenti programmi di queste pagine. Ecco un altro programma che vi mostra come utilizzare ripetutamente un piccolo numero di dati per calcolare le vostre tasse:

PROGRAMMA "TASSE"

```
LIST

10 PRINT CHR$(147)
20 PRINT : INPUT "A=";A
30 PRINT
40 FOR C=1 TO 6
50 READ T
60 PRINT A;"+";T"% =";A+A*T/100
70 NEXT C
80 RESTORE
90 GOTO 20
100 DATA 10,12.5,15,17.5,20,25
READY.
```

GLI SPRITE

Come avrete capito, visualizzare dei caratteri grafici per mezzo del comando PRINT e del comando POKE ha i suoi svantaggi: il principale consiste nella limitata risoluzione della pagina video in bassa risoluzione. Il Commodore, però, vi permette di utilizzare altri modi per visualizzare delle immagini e, uno di questi, l'uso degli sprite, produce dei disegni molto accurati che potete anche animare.

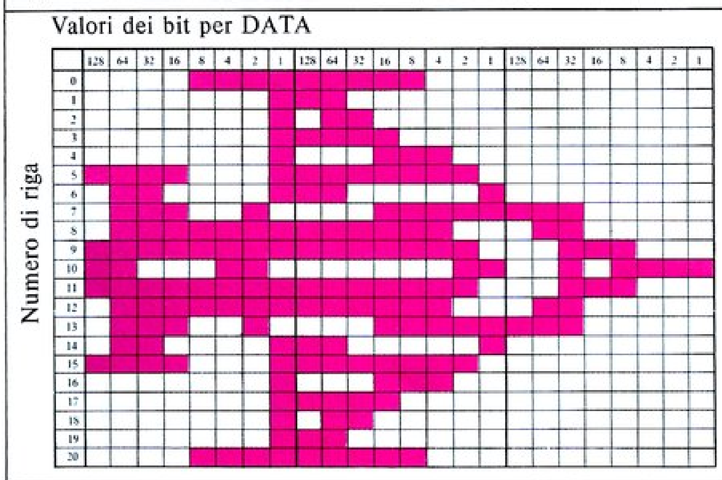
Gli sprite, detti anche MOB (Mobile Object Blocks), hanno le dimensioni di nove caratteri normali (circa) disposti in un quadrato di 3x3, e potete visualizzarne contemporaneamente fino a un massimo di otto. La forma, la dimensione e il colore degli sprite sono gestiti da uno speciale chip del Commodore, chiamato VIC (Video Interface Circuit): tutte le informazioni relative allo sprite che volete costruire devono essere memorizzate in particolari registri del chip VIC.

Costruire uno sprite

Ogni sprite è costituito da 504 puntini dello schermo (detti pixel); questi pixel sono organizzati in una griglia rettangolare con 21 righe di 24 pixel ognuna. Per costruire uno sprite, dovete innanzi tutto schizzarne il disegno su un foglio a quadretti (un quadretto=un pixel) o sulla griglia che potete trovare a pag. 59. Potete disegnare sprite di forma qualsiasi, purché essa possa essere inserita nella griglia; un possibile disegno, un aeroplano, è mostrato qui sotto, pronto per essere immesso:

USARE LA GRIGLIA PER GLI SPRITE

Lo sprite è stato disegnato su di una griglia per poterne calcolare i valori da memorizzare con DATA.



Dopo aver disegnato il vostro sprite, dovete trasferire le informazioni relative, in una forma opportuna, nella memoria del computer. Qualsiasi informazione caricata nella memoria è organizzata in byte, ognuno dei quali è costituito da otto bit. Ogni pixel di uno sprite è controllato da un bit, perciò sono necessari 24/8=3 byte per controllare ciascuna riga, riportando quali pixel devono essere accesi e quali spenti. Dal momento poi che ogni sprite è costituito da 21 righe, sono necessari 3x21=63 byte per specificarne completamente uno.

Tutte queste informazioni devono essere caricate (POKE) in locazioni di memoria consecutive all'interno del computer, in una zona della memoria in cui il chip VIC può "vederle" e a cui può accedere. Il modo più semplice per memorizzare queste informazioni è quello di utilizzare delle istruzioni DATA da cui il programma può leggerle (READ) e caricarle (POKE) in memoria. Nel programma seguente, le istruzioni DATA contengono le informazioni relative ai pixel del disegno dell'aeroplano, ma nessuna indicazione utile per visualizzare lo sprite sullo schermo:

PROGRAMMA "SPRITE SINGOLO"

```
LIST

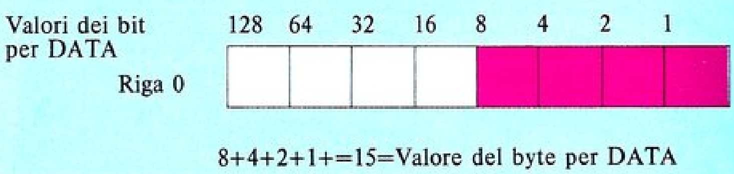
10 POKE 53280,6:POKE 53281,2
20 PRINT CHR$(147)
30 FOR C=0 TO 63
40 READ BYTE
50 POKE 832+C,BYTE
60 NEXT C
500 DATA 15,248,0,1,192,0,1,96,0
510 DATA 1,240,0,1,28,0,241,254,0
520 DATA 97,193,0,114,31,224,127,252,96
530 DATA 255,254,56,198,3,47,255,254,56
540 DATA 127,252,96,114,31,224,97,193,0
550 DATA 241,254,0,1,28,0,1,240,0
560 DATA 1,96,0,1,192,0,15,248,0,0
READY.
```

Immettere i dati relativi agli sprite

La prima cosa da notare nel programma precedente è la tecnica usata per ricavare la lista di valori da memorizzare nelle istruzioni DATA. Ecco dove intervengono i numeri e le indicazioni riportate a fianco della griglia. A partire dall'angolo superiore sinistro del disegno dello sprite, e procedendo da sinistra a destra lungo la prima riga, osservate i primi 8 pixel e i valori riportati sopra di essi. I quattro pixel più a destra sono accesi per formare la punta dell'ala, mentre gli altri quattro sono spenti:

COME SOMMARE I VALORI DEI BIT

Ogni gruppo di otto bit consecutivi in una riga di uno sprite viene codificato con un solo valore: la somma dei valori dei pixel accesi.



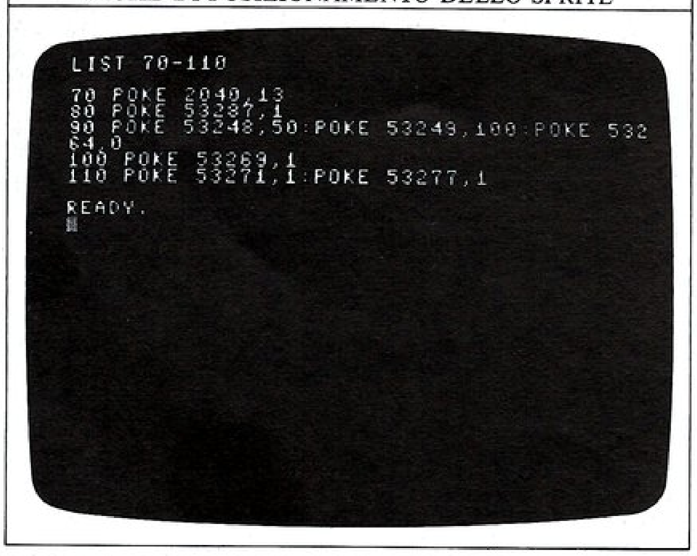
Ogni "0" indica che il pixel corrispondente non deve essere acceso, mentre un "1" rappresenta un pixel acceso. Gli otto numeri di colonna devono essere sommati per ottenere il valore del byte corrispondente. Nel nostro caso i pixel da accendere si trovano nelle colonne indicate come 8, 4, 2, 1, perciò il valore del byte ri-

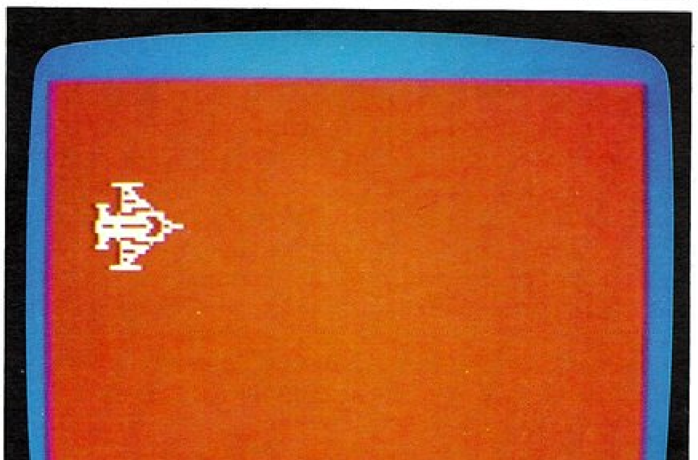
sultante è 15, cioè il primo valore presente nelle istruzioni DATA. Spostatevi sempre sugli 8 bit più a destra, attraversando tutta la tabella: otterrete così 63 valori da memorizzare in righe DATA. Il numero 63 non è particolarmente comodo da gestire da parte del computer, perciò viene aggiunto un 64esimo byte: si tratta solo di un byte di "imbottitura", non contiene cioè nessuna informazione, e vi è memorizzato zero.

Caricare gli sprite in memoria

Bisogna ora nel caricare (POKE) in memoria i valori calcolati (riga 50). Nella mappa standard della memoria del Commodore non esiste nessuna zona dedicata alla memorizzazione dei dati relativi agli sprite, perciò questo programma usa un'area di memoria normalmente dedicata alle operazioni con il registratore a cassetta; essa inizia alla locazione 828, ma questo valore non è multiplo di 64 e quindi il programma comincia a memorizzare i dati alla locazione 832 (64x13). Si ottiene così lo spazio sufficiente per 3 sprite a partire, rispettivamente, dalla locazioni 832, 896 e 960. Il ciclo FOR...NEXT tra la riga 30 e la 60 legge tutti i valori contenuti nelle istruzioni DATA e li carica in locazioni di memoria consecutive, a partire dalla 832. Se ora eseguite il programma, vedrete comparire lo sprite sullo schermo:

RIGHE DI POSIZIONAMENTO DELLO SPRITE



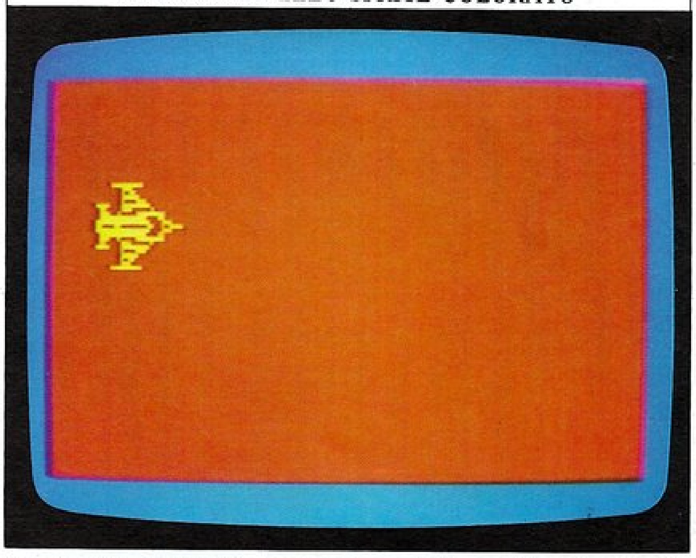


Nella memoria del Commodore esistono 8 locazioni che sono utilizzate per indicare al computer dove sono stati memorizzati i dati relativi agli sprite: un byte per ogni sprite. Dal momento che ogni byte può memorizzare un valore compreso tra 0 e 225, ci sono 256 possibili locazioni da cui cominciare a caricare i dati relativi agli sprite. Nel nostro programma, la locazione di partenza dei dati relativi allo sprite è specificata alla riga 70: essa informa il computer che si tratta del 13esimo blocco di 64 byte, cioè della locazione di memoria 832. Se modificate la riga 80 come segue, potete rendervi conto del modo con cui vengono predisposti i colori:

```
80 POKE 53287,7
```

Se eseguite il programma scoprirete che è stato cambiato il colore dello sprite:

IMMAGINE DELLO SPRITE COLORATO



La riga 90 controlla la posizione dello sprite, mentre la 100 carica (POKE) un valore nel registro che abilita e disabilita gli sprite. La riga 110, attiva due locazioni di memoria che controllano la dimensione degli sprite; i due comandi POKE eseguiti da questa riga fanno in modo che lo sprite occupi un'area dello schermo quadrupla rispetto a quella che esso richiede normalmente.

Nelle prossime due pagine, vi saranno date ulteriori informazioni su come usare questi registri di controllo; scoprirete anche come semplificare il programma precedente, perciò, prima di proseguire nella lettura, assicuratevi di aver salvato il vostro programma su nastro o su disco (a pag. 58 vi sarà detto come fare) o che esso sia tuttora presente in memoria.

Perché gli sprite non si comportano come i caratteri

Dal momento che gli sprite sono gestiti dal chip VIC, essi non si comportano allo stesso modo dei caratteri di testo. Quando eseguirete i programmi riportati in questa pagina, scoprirete di non poter cancellare gli sprite usando i tasti SHIFT e CLR e, anche quando visualizzate il listato del programma, essi rimarranno nella stessa posizione dello schermo invece di scorrere verso l'alto. Per sbarazzarvene usate i tasti RUN/STOP e RESTORE.

COME PROGRAMMARE CON GLI SPRITE

Quando state programmando con gli sprite, vi trovate a scrivere righe di programma che usano una quantità di indirizzi di POKE per il chip VIC che non potete certo ricordarvi a memoria.
Poiché gli indirizzi di questo chip sono una sequenza che va dal valore di 53248 in su, potrete aggirare il problema assegnando a una variabile il valore 53248 e quindi usandola invece del numero. Potrete fare questo con il programma delle due pagine precedenti, aggiungendo prima questa riga:

```
5 LET V=53248
```

Questo vi permette di riferirvi ad ogni registro del chip VIC in termini di V più un numero di una o due cifre. Questo rende la programmazione con gli sprite più facile da interpretare. Con questo metodo il programma di sprite delle due pagine precedenti diventerà:

PROGRAMMA "SPRITE" SEMPLIFICATO



Come colorare gli sprite

Potete visualizzare gli sprite in ognuno dei 16 colori del Commodore. Come avete visto nelle pagine precedenti si tratta solo di inserire, con un comando POKE, un codice di controllo colore che colori uno specifico sprite. I codici di colore sono gli stessi mostrati a pag. 32 per

PREDISPORRE I COLORI DEGLI SPRITE

I colori degli sprite sono controllati singolarmente da un codice colore da caricare in una locazione del chip VIC. Questa tabella fornisce sia le locazioni del VIC (V=indirizzo base del VIC), sia i codici colore.

Numero Sprite	Locazione colore VIC	Colore	Codice colore	Colore	Codice colore
0	V+39,	Nero	0	Arancione	8
1	V+40,	Bianco	1	Marrone	9
2	V+41,	Rosso	2	Rosa	10
3	V+42,	Ciano	3	Grigio scuro	11
4	V+43,	Porpora	4	Grigio	12
5	V+44,	Verde	5	Verde chiaro	13
6	V+45,	Blu	6	Azzurro	14
7	V+46,	Giallo	7	Grigio chiaro	15

l'uso con POKE. La locazione da usare con POKE dipende dal numero dello sprite: per lo sprite 0 è V+39 (come alla riga 80), per lo sprite 1 è V+40, e così via.

Come posizionare e muovere gli sprite

Se osservate ancora il programma del singolo sprite vedrete questa riga:

```
90 POKE V,50 : POKE V+16,0
```

Essa controlla la posizione video in cui comparirà lo sprite: V fissa la posizione orizzontale e V+1, la posizione verticale. Usando due soli byte per ogni sprite, potete visualizzarlo su 256 posizioni orizzontali e su 256 posizioni verticali; ma poiché ci sono più di 256 posizioni orizzontali possibili, 512 in tutto, è richiesto l'uso di un nono bit per specificare completamente la posizione di uno sprite. Invece di aggiungere un byte in più per ogni sprite, per memorizzare questa informazione, tutti e 8 i bit extra sono memorizzati in un solo registro del chip VIC, V+16. In questo byte, il bit 0 si riferisce allo sprite 0, il bit 1 allo sprite 1 e così via.

CONTROLLO DELLA POSIZIONE DEGLI SPRITE

La posizione orizzontale e verticale di ogni sprite è controllata da un diverso registro del VIC.

Numero dello Sprite	Locazione orizzontale (X) del VIC	Locazione verticale (Y) del VIC
0	V+0,	V+1,
1	V+2,	V+3,
2	V+4,	V+5,
3	V+6,	V+7,
4	V+8,	V+9,
5	V+10,	V+11,
6	V+12,	V+13,
7	V+14,	V+15,

Tutte le posizioni orizzontali non comprese tra 24 e 343, e quelle verticali non comprese tra 30 e 229, sono fuori dallo schermo. Le coordinate sono disposte così per fare in modo che gli sprite possano essere fatti scorrere in ogni direzione, dentro e fuori dallo schermo; questo significa che posizionare uno sprite sullo schermo in modo che esso si sovrapponga ad uno sfondo, può essere molto semplice: a pag. 59, è data una spiegazione di come funzionano le coordinate degli sprite. Una volta che sapete come posizionare uno sprite è facile cambiarne la posizione con un ciclo FOR...NEXT; tutto quello che rimane da fare è attivare lo sprite.
Nel programma del singolo sprite, questa operazione è eseguita alla riga 100: è necessario solo un bit per attivare e disattivare uno sprite e, anche questa volta, i bit relativi agli 8 sprite sono memorizzati in un unico registro del VIC, V+21.
Per esempio, se avete 8 sprite e volete attivarne solo alcuni ad un certo punto del programma, dovete usare un comando POKE, sommando i valori dei bit relativi:

ATTIVARE E DISATTIVARE GLI SPRITE

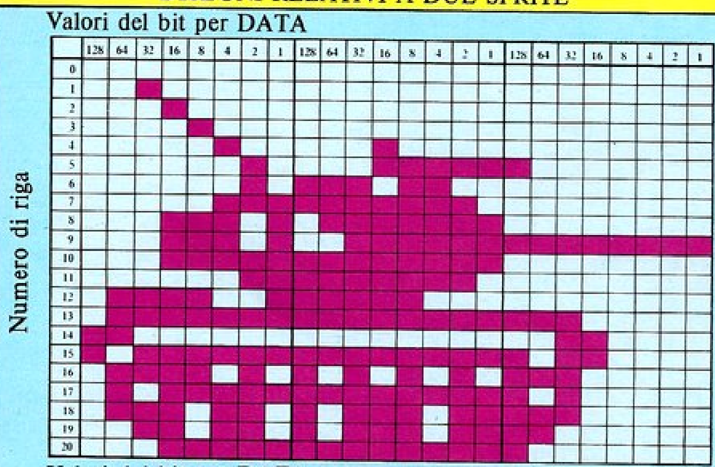
Numero Spite	128	64	32	16	8	4	2	1
Valore del bit	0	1	0	0	1	1	0	1
Stato	OFF	ON	OFF	OFF	ON	ON	OFF	ON

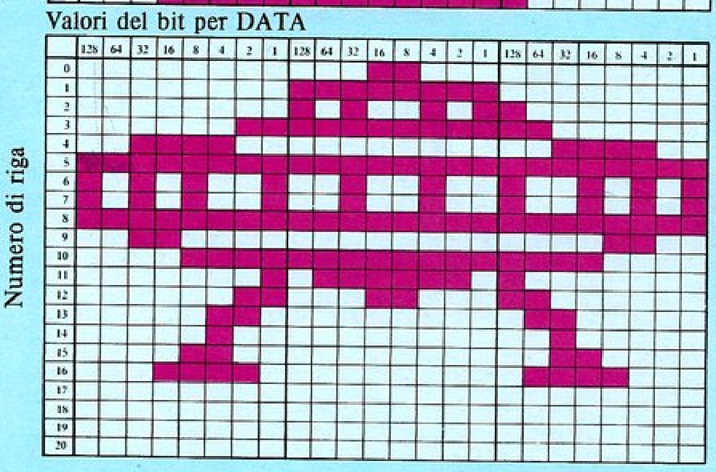
Valore totale del byte = VIC 21,77

usando la stessa tecnica potete espandere gli sprite sia orizzontalmente sia verticalmente, raddoppiando le loro dimensioni originali in entrambe le direzioni. V+29 contiene i bit che controllano l'espansione orizzontale, mentre V+23 controlla l'espansione verticale; è chiaro allora che la riga 110, nel programma dello sprite singolo, espande lo sprite 0 (il cui bit ha valore 1) in entrambe le direzioni.

Un programma di animazione con due sprite
Per mostrarvi come visualizzare più di uno sprite contemporaneamente, ecco un programma che ne crea due. Per prima cosa gli sprite sono schizzati su un foglio a quadretti:

DISEGNI RELATIVI A DUE SPRITE





Ora avete bisogno di un numero doppio di istruzioni DATA per contenere 128 byte, compresi i due byte di "imbottitura" alla fine dei dati di ogni sprite. Sono necessari anche due diversi gruppi di istruzioni per posizionare e colorare i due sprite; è inoltre possibile espandere le dimensioni verticale e orizzontale di ognuno di essi caricando il valore 3 nei registri V+23 e V+29. Il programma farà muovere i due sprite a velocità diverse:

PROGRAMMA "DUE SPRITE"

```
LIST -190

10 PRINT CHR$(147);CHR$(19);
20 POKE 53280,2:POKE 53281,12
30 PRINT CHR$(18);CHR$(31);
40 V=53248:FOR C=1 TO 15
50 PRINT "
           ";:NEXT C
60 FOR C=0 TO 127:READ BYTE
70 POKE 832+C,BYTE:NEXT C
80 X0=0:Y0=190:X1=300:Y1=75
90 D0=1:D1=-4:POKE V+21,3
100 POKE 2040,13:POKE 2041,14
110 POKE V+39,0:POKE V+40,7
120 POKE V+23,3:POKE V+29,3
130 T0=INT(X0/256):T1=INT(X1/256)
140 POKE V+16,T0+2*T1
150 POKE V,X0-T0*256:POKE V+1,Y0
160 POKE V+2,X1-T1*256:POKE V+3,Y1
170 IF X0>350 THEN X0=0
180 IF X1>300 THEN D1=-D1
190 IF X1<50 THEN D1=-D1

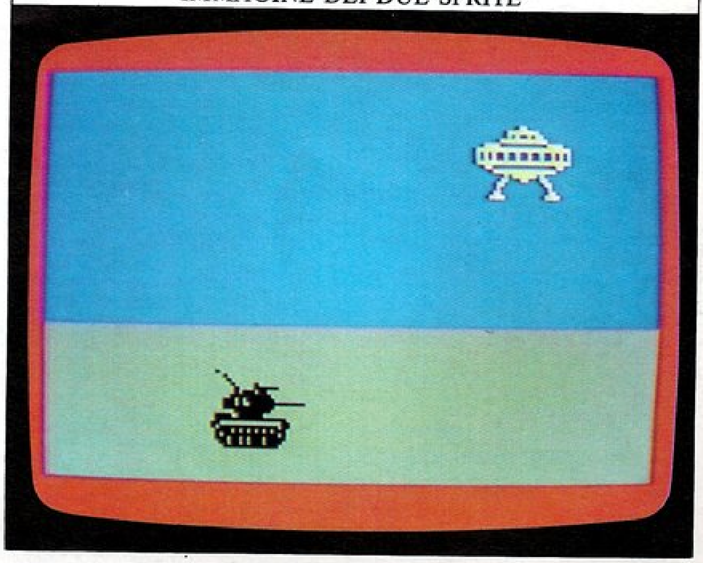
READY.
```

```
LIST 200-

200 X0=X0+D0:X1=X1+D1:GOTO 120
210 POKE V+23,3:POKE V+29,3
400 DATA 0,0,0,64,0,0,16,0,0
410 DATA 8,0,0,4,16,0,2,31,128
420 DATA 2,236,0,7,254,0,29,127,0
430 DATA 29,63,255,31,255,0,3,254,0
440 DATA 121,248,0,127,255,224,192,0,48
450 DATA 191,255,176,122,170,224,95,255,
160
460 DATA 118,187,224,54,187,192,31,255,1
28,0
470 DATA 0,24,0,0,255,0,0,165,128
480 DATA 3,255,192,60,0,60,255,255,255
490 DATA 169,36,149,169,36,149,255,255,2
55
500 DATA 48,0,12,15,255,240,1,126,128
510 DATA 3,24,192,6,0,96,4,0,32
520 DATA 12,0,48,30,0,120,0,0,0
530 DATA 0,0,0,0,0,0,0,0,0,0
READY.
```

La riga 80 controlla la posizione iniziale degli sprite; le variabili D0 e D1 (riga 90) controllano, invece, la direzione e la velocità di ognuno di essi:

IMMAGINE DEI DUE SPRITE



SUONI E EFFETTI SPECIALI

Il Commodore è equipaggiato con uno dei più sofisticati dispositivi di sintetizzazione del suono che si possono trovare in un microcomputer. Il suono può cambiare il volto dei vostri programmi, sia che si tratti di un semplice "bip" per ricordarvi di immettere i dati in corrispondenza di un'istruzione INPUT, sia nel caso di effetti sonori molto realistici, i quali aggiungono divertimento e prestigio ai vostri giochi. Il Commodore non possiede un unico comando, come ad esempio SOUND, per emettere dei suoni: usa il comando di uso generale POKE.

Il chip SID

I suoni prodotti dal Commodore sono sintetizzati da un unico circuito integrato, o chip, chiamato SID (Sound Interface Device). Il chip SID contiene tutti i circuiti necessari per realizzare tre canali sonori separati, ognuno dei quali può essere usato per produrre note musicali, rumori o effetti sonori; il computer comunica con il chip SID attraverso un gruppo di 29 locazioni di memoria, a partire dalla 54272. Nei programmi sonori seguenti, le 29 locazioni in questione saranno indicate come S+N, dove S vale 54272 (l'indirizzo di base) e N è il numero del registro SID (da 0 a 28); la locazione di memoria 54296 (la cui funzione principale è quella di controllare il volume del suono prodotto), per esempio, verrà indicata come S+24: è lo stesso sistema utilizzato nella programmazione degli sprite. Una numerazione più semplice è più facile da ricordare, con meno possibilità di errori ricopiando il testo di un programma.

Semplici programmi sonori

Per generare dei suoni dovete caricare i valori relativi alla loro intensità e alla loro variazione nel tempo ("inviluppo"), alla loro frequenza e alla loro forma d'onda: tutti questi parametri devono essere trasmessi al chip SID. Tanto per cominciare, ecco un programma che produce il suono di una sirena: un ciclo costituito da due note.

PROGRAMMA "SIRENA"

```
LIST

300 S=54272 : POKE S+24,15
310 POKE S+5,0 : POKE S+6,240
320 FOR K=1 TO 4
330 POKE S+4,33
340 POKE S,0 : POKE S+1,40
350 FOR T=1 TO 500 : NEXT
360 POKE S,0 : POKE S+1,30
370 FOR T=1 TO 500 : NEXT
380 POKE S+4,32
390 NEXT K
READY.
```

(Se eseguite questo programma e non ottenete alcun suono, controllate il volume del vostro televisore o del vostro monitor).

La riga 300 assegna alla variabile S il valore 54272 e quindi carica nel registro S+24 (il controllo di volume) il valore massimo.

L'intervallo dei valori possibili va da 0 (livello zero) a 15 (livello massimo).

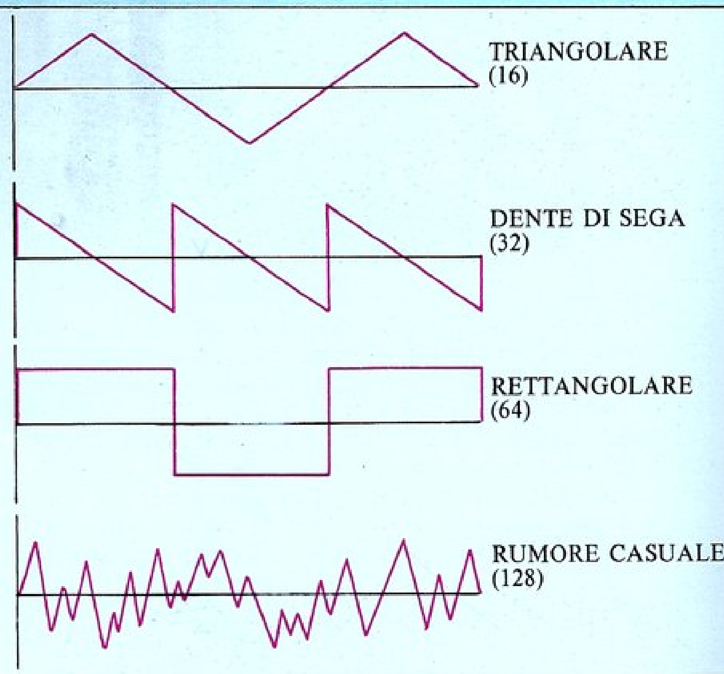
La riga 310 imposta un qualcosa detto "inviluppo", cioè la variazione dell'intensità del suono durante l'esecuzione della nota: i suoni reali non sono piatti, la loro intensità aumenta e diminuisce, ed è questa caratteristica che l'inviluppo controlla.

Alla prima metà dell'inviluppo può essere assegnato un valore qualsiasi tra 0 a 15: più alto è il valore assegnato, più lenta risulta la variazione.

Il chip SID è in grado anche di produrre suoni con quattro differenti forme d'onda:

FORME D'ONDA DEI SUONI

Una qualsiasi forma d'onda è selezionata caricando (POKE) il valore relativo nel registro che controlla la forma d'onda di ogni canale.



Ad ognuna di queste forme d'onda è assegnato un valore numerico il quale deve essere memorizzato nel registro di controllo della forma d'onda relativo al canale che volete utilizzare, per dire al chip SID quale forma d'onda volete che quel canale produca.

Le righe dalla 330 alla 380 caricano dei valori in queste locazioni. La riga 330 inizializza il canale con una forma d'onda triangolare, e la 380 termina questa operazione; dovrete sempre essere sicuri che il valore di inizializzazione sia più grande di un'unità rispetto al valore della forma d'onda selezionata.

Per poter ascoltare l'effetto prodotto dalle diverse forme d'onda, provate a modificare il programma "SIRENA" immettendo le seguenti due righe; esse variano la forma d'onda da "dente di sega" (32) a triangolare (16):

```
330 POKE S+4,17
380 POKE S+4,16
```

La frequenza o altezza delle due note della sirena è controllata dalle righe 340 e 360 (come potete notare, avete bisogno di due comandi POKE per memorizzare la frequenza). Troverete altre informazioni circa i valori della frequenza nella generazione di musica alle pagine seguenti. Le righe 350 e 370, infine, controllano la durata delle note per mezzo di un semplice ciclo di ritardo.

Costruire degli effetti sonori

Utilizzando il rumore casuale potete realizzare un certo numero di effetti sonori:

PROGRAMMA "SPARI"

```
LIST

10 S=54272 : POKE S+24,15
20 FOR K=1 TO 10
30 POKE S+4,129
40 POKE S+5,6 : POKE S+6,89
50 POKE S,0 : POKE S+1,50
60 FOR T=1 TO 65 : NEXT
70 POKE S+4,128
80 NEXT K
READY.
```

Questo programma genera il rumore prodotto da una mitragliatrice. L'inviluppo di ogni singolo sparo è controllato dalla riga 40: l'intensità del rumore casuale cresce rapidamente verso il suo massimo e quindi diminuisce abbastanza velocemente fino a raggiungere la metà del valore massimo; infine decresce lentamente fino ad annullarsi. Provate ora il seguente programma:

PROGRAMMA "CANNONE LASER"

```
LIST

100 S=54272 : POKE S+24,15
110 POKE S+5,0 : POKE S+6,249
120 POKE S+4,129
130 FOR C=0 TO 8
140 POKE S,0 : POKE S+1,44-C*4
150 FOR T=1 TO 30 : NEXT
160 NEXT C
170 POKE S+4,128
READY.
```

Il programma produce il rumore di un cannone laser, mostrando così gli effetti di una rapida variazione della frequenza da valori bassi a valori alti. Questo effetto è prodotto dalla riga 140, la quale modifica la frequenza per mezzo di un ciclo FOR...NEXT.

Infine, ecco due programmi che generano effetti sonori molto differenti. Il primo simula elettronicamente l'acutissimo canto di un uccellino, mentre il secondo emette il rumore di un motore che sta lentamente spegnendosi (premete pure RUN/STOP e RESTORE quando ne avete abbastanza!):

PROGRAMMA "USIGNOLO"

```
LIST

10 S=54272:POKE S+24,15
20 POKE S+5,0:POKE S+6,240
30 K=1
40 POKE S+4,65:POKE S+3,8
50 POKE S,0:POKE S+1,100
60 FOR T=1 TO 60:NEXT
70 POKE S+4,65:FOR C=0 TO 6
80 POKE S+1,C*15+150
90 NEXT C:POKE S+4,0
100 FOR T=1 TO 200:NEXT
110 K=K+1:IF K=2 THEN 70
120 FOR T=1 TO 250:NEXT:GOTO 10
READY.
```

PROGRAMMA "MOTORE"

```
LIST

10 Z=1
20 S=54272 : POKE S+24,10
30 POKE S+5,9 : POKE S+4,129
40 POKE S,10 : POKE S+1,20
50 FOR T=1 TO Z:NEXT
60 POKE S+5,9 : POKE S+4,17
70 POKE S,17 : POKE S+1,37
80 POKE S+4,0
90 FOR T=1 TO 10:NEXT
100 Z=1.2*Z
110 GOTO 20
READY.
```

LOCAZIONI DI MEMORIA PER IL SUONO

Per specificare un certo suono, dovete memorizzare (POKE) questi valori di controllo per ciascun canale nel chip SID.

Funzione	Canale 1	Canale 2	Canale 3
Frequenza della nota (byte meno signif.)	S+0,	S+7,	S+14,
Frequenza della nota (byte più signif.)	S+1,	S+8,	S+15,
Larghezza dell'onda rettang. (byte meno signif.)	S+2,	S+9,	S+16,
Larghezza dell'onda rettang. (byte più signif.)	S+3,	S+10,	S+17,
Forma d'onda/controllo canale principale	S+4,	S+11,	S+18,
Forma dell'inviluppo (attacco/caduta)	S+5,	S+12,	S+19,
Forma dell'inviluppo (mantenimento/rilascio)	S+6,	S+13,	S+20,
Filtri/controllo principale di volume	S+24,	S+24,	S+24,

NOTE, ACCORDI E MUSICA

Tutti i programmi sonori usati finora si servivano del canale 1, ma avreste potuto facilmente utilizzare il canale 2 o il 3. Il Commodore non è però limitato ad usare un solo canale sonoro alla volta; con il programma adatto potete ottenere anche degli accordi (un certo numero di note suonate contemporaneamente), e questi ultimi possono essere raggruppati per ottenere un'armonia. Per suonare una semplice sequenza di note, usando un solo canale, dovete immettere una lista di righe (POKE) che memorizzano i valori relativi alle note nella coppia di registri che controllano la frequenza. Ecco un programma di esempio che utilizza il canale 1:

PROGRAMMA "MELODIA SUL CANALE 1"

```
LIST

100 S=54272 : POKE S+24,15
110 POKE S+5,0 : POKE S+6,160
120 POKE S+4,33
130 POKE S,195 : POKE S+1,16
140 FOR T=1 TO 160 : NEXT T
150 POKE S+4,32
160 FOR T=1 TO 20 : NEXT T
170 POKE S+4,33
180 POKE S,195 : POKE S+1,16
190 FOR T=1 TO 80 : NEXT T
200 POKE S,209 : POKE S+1,18
210 FOR T=1 TO 320 : NEXT T
220 POKE S,195 : POKE S+1,16
230 FOR T=1 TO 320 : NEXT T
240 POKE S,96 : POKE S+1,22
250 FOR T=1 TO 320 : NEXT T
260 POKE S,30 : POKE S+1,21
270 FOR T=1 TO 320 : NEXT T
280 POKE S+4,32
READY.
```

Questo metodo funziona bene, ma potrete scoprire che richiede un programma di molte righe per ottenere una melodia di una certa lunghezza. Una soluzione più efficiente consiste nel memorizzare i valori della frequenza in istruzioni DATA e quindi scrivere un breve programma che lêgge e suona le note. Ecco il programma precedente con una piccola modifica:

PROGRAMMA "MELODIA" CON ISTRUZIONI **DATA**

```
LIST

300 S=54272 : POKE S+24,15
310 POKE S+5,0 : POKE S+6,160
320 READ L,H,D
330 IF H=0 THEN FOR T=1 TO 20 : NEXT T :
 GOTO 320
340 IF H<0 THEN END
350 POKE S+0,L : POKE S+1,H
360 POKE S+4,33
370 FOR T=1 TO D : NEXT T
380 POKE S+4,32
390 GOTO 320
400 DATA 195,16,160,0,0,0,195,16,80
410 DATA 209,18,320,195,16,320
420 DATA 96,22,320,30,21,320,0,-1,0
READY.
```

Ogni nota richiede di memorizzare in un'istruzione DATA tre valori: due per la sua frequenza e uno per la sua durata. Se il byte alto della frequenza è 0, il programma genera una pausa mentre, se esso è negativo, il programma termina (senza visualizzare alcun numero di righe).
Una volta che avrete immesso questo programma, l'ottenere una melodia qualsiasi si riduce alla conversione di ogni nota nei suoi valori (più significativo e meno significativo) di controllo della frequenza, e alla memorizzazione di questi ultimi in una istruzione DATA, assieme con un valore per la durata.

Come ottenere degli accordi
Utilizzando solo il canale 1 potete suonare delle singole note, ma usando tutti e tre i canali potete addirittura suonare degli accordi.

SEMPLICE ACCORDO

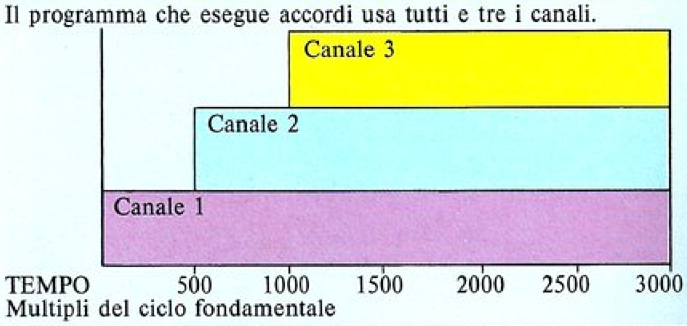
```
LIST

10 S=54272 : POKE S+24,15
20 POKE S+5,0 : POKE S+6,160
30 POKE S+12,0 : POKE S+13,160
40 POKE S+19,0 : POKE S+20,160
50 POKE S+4,17
60 POKE S+0,98 : POKE S+1,8
70 FOR T=1 TO 500 : NEXT T
80 POKE S+11,17
90 POKE S+7,143 : POKE S+8,10
100 FOR T=1 TO 500 : NEXT T
110 POKE S+18,17
120 POKE S+14,143 : POKE S+15,12
130 FOR T=1 TO 2000 : NEXT T
140 POKE S+4,16 : POKE S+11,16 : POKE S+
18,16
READY.
```

Eseguite il programma e ascoltatene i risultati.
L'accordo prodotto è costituito da tre note emesse contemporaneamente, una per ogni canale; potete dunque eseguire qualsiasi motivo che contenga accordi costituiti da un massimo di tre note.
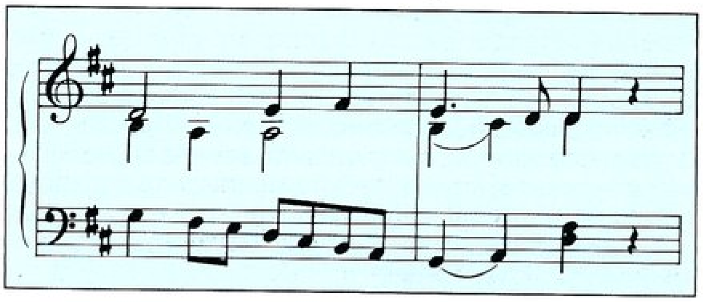
UTILIZZARE CONTEMPORANEAMENTE I TRE CANALI

Il programma che esegue accordi usa tutti e tre i canali.



Memorizzare i pentagrammi
Il programma seguente è una versione che genera accordi del programma che utilizzava l'istruzione DATA per memorizzare i valori relativi alle frequenze delle

note. Il pentagramma seguente mostra le note generate da questo programma; quando lo proverete, ricordate che le note "collegate" non hanno nessuna soluzione di continuità e dovete trattarle come una nota unica, la cui durata è la somma delle durate delle singole note.



Come funziona il programma per generare musica

Quando generate musica con il computer, dovete ricordare una cosa molto importante: una volta che il chip SID è stato programmato per generare un suono su un determinato canale, continuerà a farlo fino a che non lo annulliate, o fino a quando non ne programmate uno nuovo su quel canale. Nel seguente programma per la generazione di musica, le istruzioni DATA contengono ancora i valori della frequenza delle note e della loro durata ma, questa volta, esse controllano anche il canale selezionato. Se guardate alla riga 130, vedrete una serie di nove numeri. Il primo valore di ogni gruppo di tre è il numero del canale, ed è seguito dai due valori relativi alla frequenza di una delle note del primo accordo. Se ora osservate la riga 140, scoprirete che il primo valore è 0. La riga 70 dice al computer che, dopo aver letto uno 0, esso deve interpretare il valore seguente come la lunghezza di un ciclo di ritardo; nel nostro caso il valore successivo è 160, ciò significa che il computer eseguirà FOR T=1 TO 160 prima di leggere il prossimo valore memorizzato nell'istruzione DATA, il quale reinizializza il canale 2.

PROGRAMMA "PENTAGRAMMA"

```
LIST -120

10 S=54272 : POKE S+24,15
20 POKE S+5,0 : POKE S+6,240
30 POKE S+12,0 : POKE S+13,240
40 POKE S+19,0:POKE S+20,240
50 POKE S+4,33 : POKE S+11,33 : POKE S+1
8,33
60 READ C
70 IF C=0 THEN READ D : FOR T=1 TO D : N
EXT T : GOTO 60
80 IF C<0 THEN 120
90 READ L,H
100 POKE S+(C*7)-7,L : POKE S+(C*7)-6,H
110 GOTO 50
120 POKE S+4,32 : POKE S+11,32 : POKE S+
18,32 : END
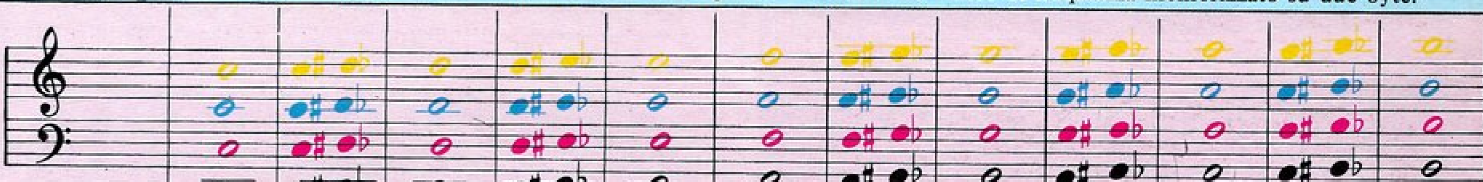
READY.
█
```

```
LIST 130-

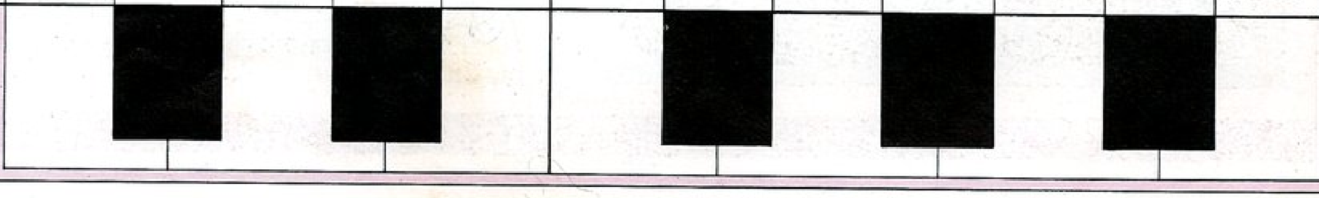
130 DATA 1,209,18,2,210,15,3,143,12
140 DATA 0,160,2,24,14,3,218,11,0,80
150 DATA 3,143,10,0,80,1,30,21,2,24,14,3
,48,11,3,104,9
160 DATA 0,80,3,225,8,0,80,1,180,23
170 DATA 3,233,7,0,80,3,12,7,0,80
180 DATA 1,30,21,2,210,15,3,71,6,0,160°
190 DATA 2,194,17,3,12,7,0,80
200 DATA 1,209,18,0,80,1,218,11,2,24,14
210 DATA 3,104,9,0,320
220 DATA -1
READY.
█
```

SCEGLIERE UN VALORE DI FREQUENZA

Valore della frequenza delle note contenute in sette ottave: ogni nota richiede un valore di frequenza memorizzato su due byte.



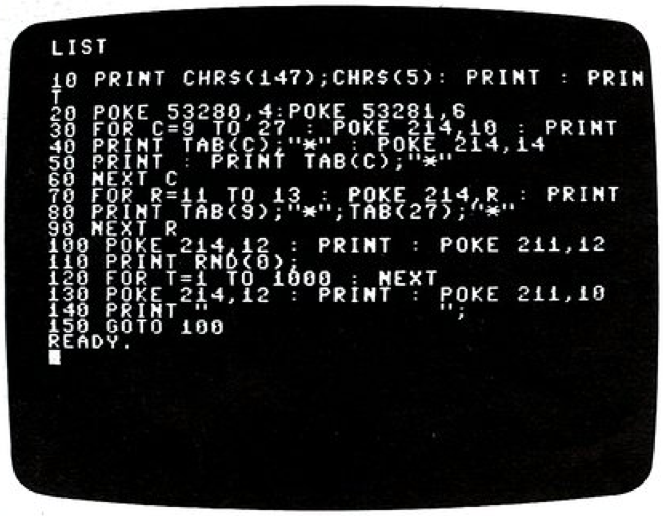
NOTA	DO	DO#REb	RE	RE#MIb	MI	FA	FA#SOLb	SOL	SOL#LAl	LA	LA#SIb	SI#DOb
7	134 24	142 18	150 132	159 120	168 243	178 255	189 164	200 235	212 221	225 133	238 238	253 35
6	67 12	71 9	75 66	79 188	84 122	89 127	94 210	100 117	106 110	112 195	119 119	126 146
5	33 134	35 132	37 161	39 222	42 61	44 192	47 105	50 59	53 55	56 97	59 188	63 73
4	16 195	17 194	18 209	19 239	21 30	22 96	23 180	25 29	26 156	28 49	29 222	31 164
3	8 98	8 225	9 104	9 247	10 143	11 48	11 218	12 143	13 78	14 24	14 239	15 210
2	4 49	4 113	4 180	4 252	5 72	5 152	5 237	6 71	6 167	7 12	7 119	7 233
1	2 24	2 56	2 90	2 126	2 164	2 204	2 247	3 36	3 33	3 134	3 188	3 245
0	1 12	1 28	1 45	1 63	1 82	1 102	1 123	1 146	1 170	1 195	1 222	1 250

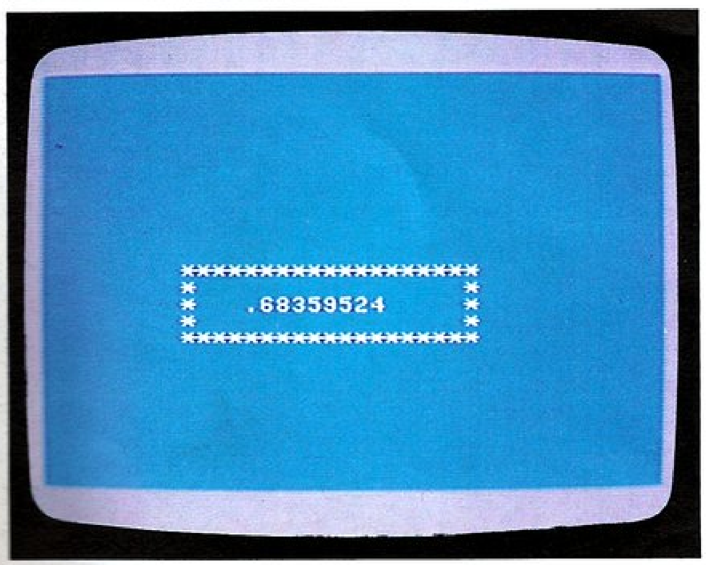


PROGRAMMI IMPREVEDIBILI

Benché il computer, generalmente, utilizzi informazioni ben precise, facendo esattamente quello che voi gli dite di fare, la maggior parte dei video giochi è basata su scelte i cui risultati non sono determinabili a priori. Se volete far accadere qualcosa ad un istante non determinato, oppure se devono essere tirati dei dadi o una moneta, non potete specificare al computer quali risultati produrre ogni volta: la componente di casualità scomparirebbe. È possibile introdurre il fattore casualità nel computer utilizzando la funzione RND. Avete già incontrato questa parola chiave: è stata usata, per esempio, per produrre una serie di numeri casuali nel programma di test matematici alle pagg. 30 e 31. RND è una abbreviazione di RaNDom (casuale); essa permette di generare dei numeri casuali minori di un valore massimo che potete scegliere, e voi potete utilizzare questi numeri per ottenere delle sequenze del tutto imprevedibili. Il programma seguente usa RND per visualizzare una serie di numeri casuali (i numeri selezionati dal programma sono tutti minori di 1):

PROGRAMMA "NUMERI CASUALI"





Nella riga 110 è utilizzata appunto la funzione RND(0) per generare numeri casuali tra 0 e 0.99999999, e le righe dalla 30 alla 90 costruiscono una cornice di asterischi che circonda il numero generato. I numeri molto piccoli conterranno il simbolo "E" che avete già incontrato a pag. 19. Normalmente, non appena un numero viene visualizzato, esso cancella il numero precedente sovrapponendosi ad esso; quando, però, viene generato un numero del tipo di E-4, esso non viene cancellato automaticamente. La riga 140 deve essere aggiunta al programma proprio per ovviare a questo inconveniente.

Benché RND venga definita come una funzione casuale, essa non lo è esattamente; può essere definita più correttamente come funzione "pseudocasuale", e produce risultati che seguono un certo schema di ripetitività. Questa particolare sequenza ripetitiva è determinata dal tempo trascorso dal momento dell'accensione del vostro computer; siccome questo intervallo di tempo può assumere un valore qualsiasi, i numeri prodotti possono essere considerati completamente casuali per la maggior parte delle applicazioni.

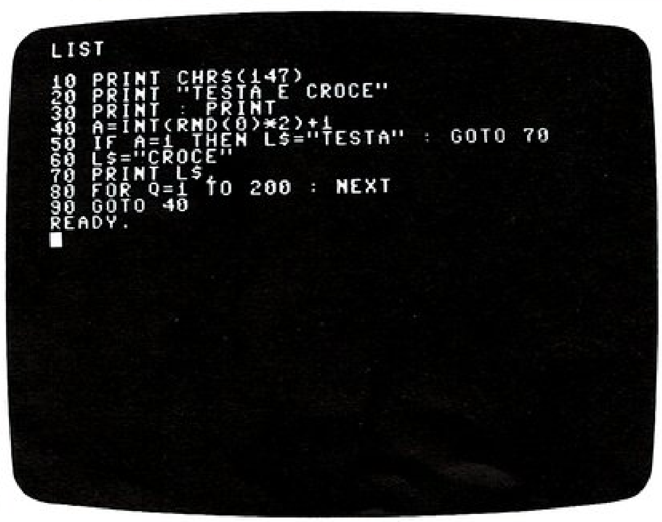
Generare dei numeri interi casuali

Se voi sostituite la riga 110 con:

```
110 PRINT INT(RND(0)*10);
```

ed eseguite nuovamente il programma; scoprirete che i numeri visualizzati sono differenti da quelli della versione precedente: non possiedono più alcuna parte decimale; ora il programma genera dei numeri casuali compresi tra 0 e 9 (estremi inclusi), i quali sono utilizzati molto spesso nei programmi. INT, che avete già incontrato a pag. 29, arrotonda i numeri casuali generati alla loro parte intera. Grazie all'uso di RND risulta molto semplice simulare con il computer il lancio dei dadi o quello di una moneta; ecco appunto un programma che utilizza dei numeri casuali per simulare il lancio di una moneta:

PROGRAMMA "TESTA O CROCE"



Siccome il lancio di una moneta può fornire solo due valori (testa o croce), la riga 40 produce dei numeri casuali che possono assumere solo il valore 1 o 2: 1 rappresenta "testa" e 2 rappresenta "croce". La variabile L$ contiene il risultato dell'ultimo lancio eseguito; se A vale 1, a L$ viene assegnato il valore "TESTA" e la riga 60 non viene eseguita. Se A vale 2, il computer decide che la condizione alla riga 50 è falsa e perciò assegna a L$ il valore "CROCE" (riga 60). Il ciclo FOR...NEXT (riga 80), che utilizza la variabile Q, realizza una breve pausa tra ogni lancio:

VISUALIZZAZIONE DEL PROGRAMMA "TESTA O CROCE"

```
TESTA E CROCE

TESTA      CROCE      CROCE      TESTA
CROCE      TESTA      TESTA      CROCE
TESTA      CROCE      TESTA      CROCE
CROCE      TESTA      CROCE      TESTA
CROCE      TESTA      CROCE      TESTA
CROCE      TESTA      CROCE      CROCE
CROCE      CROCE      TESTA      CROCE
TESTA      CROCE      CROCE      TESTA
CROCE      CROCE      CROCE      CROCE
TESTA      TESTA      CROCE      TESTA
TESTA      CROCE      TESTA      CROCE
TESTA      CROCE      CROCE      TESTA
CROCE      CROCE      TESTA      CROCE
CROCE      CROCE      CROCE      TESTA
TESTA      CROCE      CROCE      TESTA
CROCE      CROCE      CROCE      CROCE
TESTA      TESTA      CROCE      TESTA
```

La distribuzione di probabilità in un programma casuale
È possibile scrivere un programma che mostra la distribuzione di probabilità di RND. Se usate RND per lanciare (elettronicamente) una moneta 100 volte, otterrete circa 50 volte "testa" e 50 "croce" a ogni esecuzione. Potete provare veramente, per convincervi di questo fatto: immettete il programma seguente modificando quello precedente. Quando lo eseguirete, visualizzerà il risultato finale dei lanci, per mostrarvi come il numero di "teste" sia pressocché uguale a quello delle "croci":

PROGRAMMA "TEST SU RND"

```
LIST

10 PRINT CHR$(147)
20 PRINT TAB(12);"TESTA E CROCE"
30 FOR K=1 TO 100
40 A=INT(RND(0)*2)+1 : N=N+1
50 IF A=1 THEN L$="TESTA" : GOTO 70
60 L$="CROCE "
70 POKE 214,5 : PRINT : POKE 211,10
80 PRINT "QUESTO LANCIO - ";L$ :PRINT
90 PRINT TAB(5);"TESTA";TAB(28);"CROCE"
100 IF A=1 THEN T=T+1 : GOTO 120
110 C=C+1
120 POKE 214,9 : PRINT : POKE 211,5
130 PRINT TAB(6);T;TAB(29);C : PRINT
140 PRINT TAB(10);"NUMERO LANCI =";N
150 FOR Z=1 TO 500 : NEXT
160 NEXT K
READY.
```

Usare la casualità nei programmi di grafica
Potete ottenere degli effetti spettacolari utilizzando la funzione RND nei programmi di grafica, in modo che il computer possa visualizzare dei simboli grafici casuali in posizioni altrettanto casuali sullo schermo. Se poi ripetete questo procedimento un certo numero di volte, per mezzo di un ciclo FOR...NEXT, potete ottenere un disegno che cambia ogni volta che eseguite il programma. Il programma seguente utilizza proprio questa tecnica. La riga 40 seleziona un colore casuale per il testo e le righe 50 e 60 preparano due coordinate, X e Y, anch'esse imprevedibili:

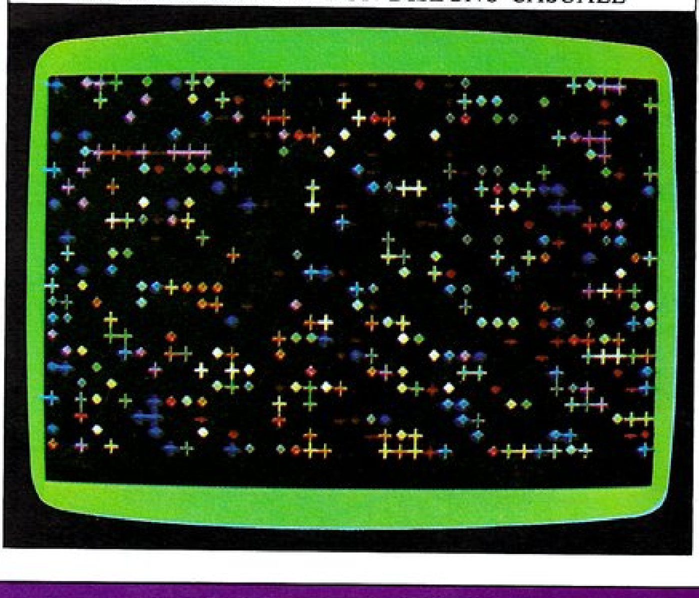
PROGRAMMA "DISEGNO CASUALE"

```
LIST

10 PRINT CHR$(147);CHR$(5)
20 POKE 53280,5:POKE 53281,0
30 FOR N=1 TO 1000
40 COL=RND(0)*15
50 X=INT(RND(0)*40)
60 Y=INT(RND(0)*23)
70 POKE 1024+Y*40+X,90+RND(1)*2:POKE 552
96+Y*40+X,COL
80 NEXT N
READY.
```

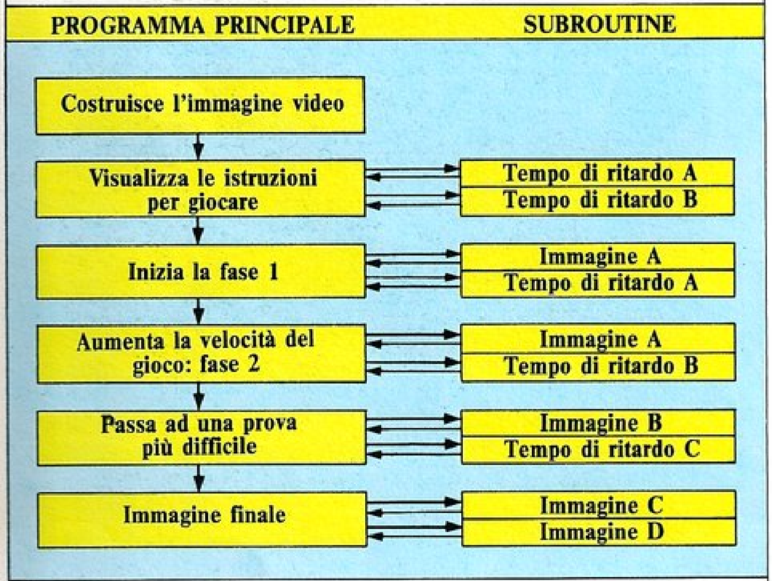
La riga 70 sembra molto più complicata di quanto non lo sia in realtà: usando la formula che potete trovare a pag. 36, essa visualizza un carattere (in un determinato colore) nella posizione X, Y dello schermo. In questo caso può trattarsi di un rombo o di una crocetta; il carattere da visualizzare viene scelto ogni volta dalla formula 90+RND(1)*2, la quale fornisce il codice dello stesso. Potete utilizzare questo programma come base per ottenere un elevato numero di disegni differenti, modificando solo la selezione dei caratteri e dei colori:

VISUALIZZAZIONE DI UN DISEGNO CASUALE



COME SCRIVERE LE SUBROUTINE

Vi capiterà spesso di dover usare più e più volte lo stesso piccolo numero di righe di programma per eseguire dei calcoli o per visualizzare lo stesso gruppo di caratteri sullo schermo. Per evitare di dover riscrivere ogni volta le stesse istruzioni (occupando così troppa memoria), dovreste usare frequentemente il comando GOTO per saltare ad una certa parte del programma. Questo comando è però considerato con sospetto da molti programmatori: usarlo in modo dissennato può portare a programmi simili a intricati labirinti, che sono praticamente impossibili da capire e da correggere.
Il programma più facile da interpretare e correggere è quello sistematicamente suddiviso in blocchi o moduli, ognuno dei quali può essere provato indipendentemente dagli altri in caso di problemi. Se date un'occhiata al listato di un buon video gioco pubblicato da qualche rivista, per esempio, scoprirete che esso è costruito più o meno in questo modo:



Come usare una subroutine
Nel programmare il Commodore, le parti di programma usate più spesso possono essere scritte a parte come subroutine, e richiamate col comando GOSUB. Questo comando permette di "saltare fuori" dal programma principale, entrare in una routine e poi tornare al programma principale; esso ha la seguente forma:

```
50 GOSUB 500
```

In questo caso il programma principale segue normalmente il suo corso fino a quando non raggiunge la riga 50, che lo costringe a saltare alla subroutine che inizia alla riga 500; dopo che tale subroutine è stata eseguita, il controllo ritorna automaticamente al programma principale, che riprende dalla riga 60, quella seguente il punto in cui era stato interrotto. Ogni subroutine deve terminare con la parola chiave RETURN; senza di essa, il computer non potrà ritornare al programma principale. Potete usare GOSUB in quasi tutti i programmi in cui occorra ripetere una operazione. Il programma seguente produce una tabella di conversione di temperatura, usando tre unità di misura: gradi Centigradi, gradi Fahrenheit e Kelvin. La subroutine dalla riga 90 alla riga 110 fa visualizzare al computer una linea della tabella e quindi cede il controllo alla riga 70. Il comando END alla riga 80 impedisce al programma principale di proseguire sulle routine. Se omettete END, esso raggiungerà l'istruzione RETURN alla riga 110: verrà allora emesso un messaggio di errore in quanto il computer ha rilevato un RETURN senza aver eseguito il comando GOSUB corrispondente. Potete anche notare che il comando GOSUB si trova all'interno di un ciclo FOR...NEXT e, perciò, la subroutine viene richiamata ogni volta che la temperatura in gradi centigradi è incrementata dall'istruzione NEXT. In questo ciclo è presente una nuova parola chiave, STEP: essa permette di incrementare C di 10 in 10 anziché di 1 in 1. Non è necessario che il valore di STEP sia un numero intero (può perfino essere negativo):

PROGRAMMA DI CONVERSIONE DI TEMPERATURA

```
LIST

10 PRINT CHR$(147);CHR$(31)
20 POKE 53280,8 : POKE 53281,3
30 PRINT : PRINT TAB(6);"C";TAB(18);"F";
TAB(31);"K"
40 PRINT "----------------------------------------"
50 FOR C=-40 TO 120 STEP 10
60 GOSUB 90
70 NEXT C
80 END
90 PRINT TAB(4);C;TAB(16);C*9/5+32;TAB(2
9);C+273
100 FOR A=1 TO 200 : NEXT A
110 RETURN
READY.
```

C	F	K
-40	-40	233
-30	-22	243
-20	-4	253
-10	14	263
0	32	273
10	50	283
20	68	293
30	86	303
40	104	313
50	122	323
60	140	333
70	158	343
80	176	353
90	194	363
100	212	373
110	230	383
120	248	393

In questo programma, in realtà, la routine non fa risparmiare dello spazio, tuttavia, se volete ampliare il programma per fargli svolgere altre funzioni, essa potrà ancora essere richiamata tutte le volte che volete, permettendovi di risparmiare sia spazio che memoria.

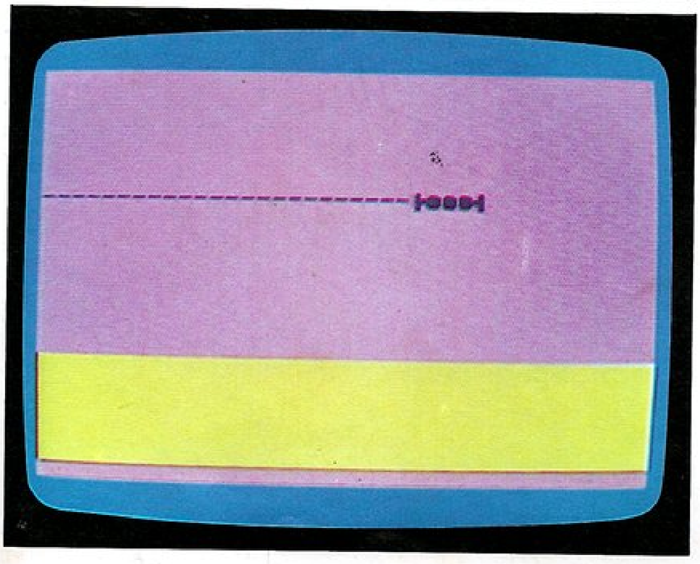
Costruire i "menù" con GOSUB

In molti programmi vi vien proposto un menù iniziale, cioè una lista delle scelte che potete operare.
Questo menù è molto spesso costruito tramite dei GOSUB.
Quando effettuate la vostra scelta, il programma salta alla subroutine relativa e costruisce la schermata del gioco o della funzione che volete eseguire.
Ecco un semplice listato che mostra come potete fare tutto ciò: esso è in grado di visualizzare, mediante una subroutine, una immagine video i cui colori dipendono dal dato che avete immesso alla riga 20. In questo programma un gruppo di caratteri ottenibili da tastiera si muove sullo schermo.
Anche usandolo all'interno di un gioco più complesso, potrete richiamare la subroutine tutte le volte che vorrete:

PROGRAMMA A MENÙ

```
LIST

10 PRINT CHR$(147)
20 INPUT "1 / 2";A
30 PRINT CHR$(147) : IF A=1 THEN G=158 :
 S=4 : C=144 : GOSUB 200
40 IF A=2 THEN G=28 : S=5 : C=28 : GOSUB
 200
50 FOR X=0 TO 34
60 POKE 214,6 : PRINT : POKE 211,X : PRI
NT "-├●●●┤"
70 FOR M=1 TO 50: NEXT M
80 NEXT X
90 POKE 214,6:PRINT:POKE 211,34:PRINT"
 "
100 GOTO 50
200 POKE 53280,6: POKE 53281,S
210 POKE 214,16 : PRINT
220 PRINT CHR$(G) : FOR Y=1 TO 280
230 PRINT "█";: NEXT Y
240 PRINT CHR$(C); : RETURN
READY.
█
```



GOSUB è un comando utile per molti programmi per video giochi, in cui viene suonato un motivetto per tutta la durata del gioco. Invece di inserire ogni volta le istruzioni per l'esecuzione delle varie note, potete scriverle una volta per tutte e, quindi, immettere un comando GOSUB seguito dall'appropriato numero di riga ogni volta che volete eseguire il motivetto.

Usare GOSUB per l'animazione

Come avete scoperto non appena avete avuto a che fare con semplici disegni, costruire un programma per animare figure può rivelarsi molto lungo, in particolare se volete animare molti caratteri. GOSUB è particolarmente utile nei programmi per l'animazione:

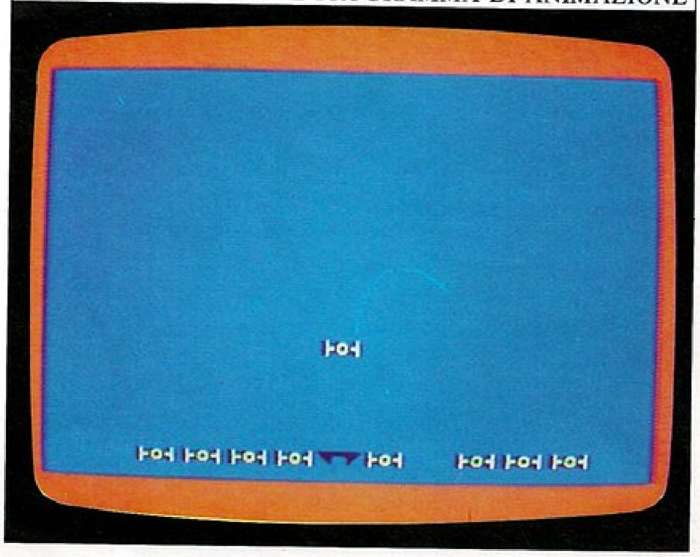
PROGRAMMA DI ANIMAZIONE CON **GOSUB**

```
LIST

10 PRINT CHR$(147)
20 POKE 53280,2 : POKE 53281,6
30 POKE 214,23 : PRINT : POKE 211,18
40 PRINT CHR$(144);CHR$(127);CHR$(184);C
HR$(169);
50 X=INT(RND(1)*13)*3
60 PRINT CHR$(158) : FOR Y=1 TO 22
70 POKE 214,Y-1 : PRINT : POKE 211,X
80 PRINT " " : POKE 211,X
90 PRINT CHR$(171);CHR$(119);CHR$(179);
100 IF X=18 THEN IF Y=22 THEN GOSUB 500
110 FOR T=1 TO 5:NEXT T
120 NEXT Y : GOTO 50
500 FOR T=1 TO 20
510 POKE 214,22 : PRINT : POKE 211,18
520 PRINT CHR$(28);CHR$(127);CHR$(184);C
HR$(169);
530 POKE 214,22 : PRINT : POKE 211,18
540 PRINT CHR$(144);CHR$(127);CHR$(184);
CHR$(169);
550 NEXT T : RETURN
READY.
█
```

Ecco un programma che visualizza un bersaglio (tre simboli grafici) terrestre e un certo numero di astronavi che precipitano da posizioni casuali dello schermo. Se una di esse colpisce il bersaglio la riga 100 richiama la subroutine alla riga 500, la quale produce una serie di flash in diversi colori prima che il gioco riprenda nuovamente:

VISUALIZZAZIONE DEL PROGRAMMA DI ANIMAZIONE



CONSIGLI E SUGGERIMENTI

Mentre stavate imparando a programmare il vostro Commodore, vi sarete certamente imbattuti in un certo numero di metodi che vi permettevano di migliorare la vostra abilità provando e riprovando. Esistono tuttavia altri metodi per risparmiare tempo o per evidenziare problemi che, benché semplici ed evidenti, non sono necessariamente ovvi. In queste due pagine imparerete dei trucchi che vi aiuteranno a produrre programmi ben organizzati e privi di errori.

Usare REM come un indicatore o una maschera
Siccome il comando REM fa in modo che il computer ignori tutto ciò che lo segue in una certa riga, può essere usato per evidenziare e provare parti di un programma. A pag. 20 avete imparato ad usare REM come prima istruzione di un programma per indicare lo scopo del programma stesso; potete però usare REM lungo tutto il listato per ricordarvi come funziona ogni sua singola parte e, più lungo è il programma, più utile è l'uso di questo comando. Tuttavia, quando un programma diventa veramente lungo, può essere difficile individuare le righe di REM in mezzo a tutte le altre. Un modo per attirare l'attenzione su di esse è quello di far seguire ad ogni REM dei simboli che si differenzino nettamente da tutti gli altri usati nel programma. Ecco un modo per fare ciò:

PROGRAMMA CON RIGHE DI **REM**

```
LIST

2 REM ************************
4 REM GRAFICO DELLA TEMPERATURA
6 REM ************************
8 REM
10 PRINT CHR$(147);CHR$(31)
20 POKE 53280,10 : POKE 53281,3
30 PRINT : PRINT TAB(6);"C";TAB(18);"F";
TAB(31);"K"
40 PRINT "------------------------------
-----------"
50 FOR C=-40 TO 120 STEP 10
60 GOSUB 90
70 NEXT C
80 END
90 PRINT TAB(4);C;TAB(16);C*9/5+32;TAB(2
9);C+273
100 FOR A=1 TO 200 : NEXT A
110 RETURN
READY.
█
```

Quando leggete il testo di questo programma, le istruzioni REM risultano immediatamente riconoscibili. REM può essere usato anche durante lo sviluppo del programma. Capiterà spesso di voler provare solo parti di un programma per vedere cosa succede eliminando certe istruzioni (forse perché il resto richiede un tempo di esecuzione troppo lungo oppure emette dei suoni che non volete ascoltare).
Potete evitare una parte del programma utilizzando GOTO oppure RUN seguito da un numero di riga, ma questi metodi non vi aiuteranno se volete eliminare solo poche righe nel mezzo dello stesso. L'unico modo per risolvere questo problema senza dover riordinare, cancellare o perdere tutte le righe, è quello di inserire un REM all'inizio di ogni riga che non vi interessa. Questa operazione "disabiliterà" quelle righe, dal momento che il computer ignora tutto ciò che segue REM. Ecco un programma a cui è stato applicato questo trucco:

RIGHE MASCHERATE CON **REM**

```
LIST

10 POKE 53280,2:POKE 53281,2
20 PRINT CHR$(147);CHR$(5)
30 REM PRINT " A"," A↑2"," A↑3"
40 REM PRINT "--------------------------
-" : PRINT
50 FOR N=1 TO 10
60 PRINT N,N*N,N*N*N
70 NEXT N
READY.
█
```

Come controllare i cicli innestati
Quando usate un certo numero di cicli in un programma, è facile che questi si aggroviglino in modo tale che il programma non possa produrre il risultato che vi aspettavate. Esiste però un metodo per controllare se i vari cicli si innestano senza sovrapporsi.
Potete fare questo controllo unendo l'inizio e la fine di ogni ciclo con una linea. Ecco un programma che contiene dei cicli innestati, il quale vi mostra come queste linee di connessione dovrebbero presentarsi, una dentro l'altra. Ci sono tre cicli FOR...NEXT, due dei quali contenuti nel ciclo principale dalla riga 20 alla 90:

PROGRAMMA CON CICLI INNESTATI

```
LIST

  10 PRINT CHR$(147)
┌─20 FOR Y=0 TO 20
│┌30 FOR X=0 TO 39
││40 PRINT CHR$(144);"  █";
││50 POKE 55296+Y*40+X,X
│└60 NEXT X
│ 70 FOR T=1 TO 50
│[80 NEXT T
└─90 NEXT Y
READY.
█
```

Quando avrete connesso ogni FOR con il relativo NEXT, scoprirete che queste linee non si sovrappongono mai. Se ciò dovesse accadere, significa che avete innestato i cicli in modo errato ed è molto probabile che il vostro programma non funzioni correttamente. Naturalmente non potete disegnare queste linee direttamente sullo schermo, ma potete utilizzare i vostri appunti o la stampa del listato.

Utili tecniche di debugging

Benché il Commodore possieda un vasto repertorio di messaggi di errore che vi avvertiranno di qualsiasi riga errata presente nel programma, spesso un programma non darà luogo a nessun messaggio di errore ma produrrà un risultato completamente diverso da quello che avevate preventivato. Come potete fare per scoprire la causa del problema?

Come avete appena visto, potete usare REM per mascherare parte del programma, o potete collegare i cicli per controllare che essi siano innestati correttamente. Se però tutto questo non vi aiuta, potete, molte volte, rintracciare l'errore assegnando a ogni variabile del programma un valore ben definito, invece di permetterle di assumerne uno tra molti.

Immaginate di avere un programma grafico che utilizza la funzione RND per costruire un disegno per mezzo di un ciclo. Se esso non funziona nel modo che vi aspettate vi conviene sostituire RND con un numero predefinito; calcolare poi quali risultati devono essere prodotti dal programma in quel caso e, quindi, eliminare le righe di inizio e di fine del ciclo (potete usare REM a questo scopo). Se il risultato della successiva esecuzione non è quello calcolato a tavolino, l'immagine prodotta vi potrà dare sicuramente un'idea del punto in cui il programma è sbagliato:

PROGRAMMA MODIFICATO PER LA PROVA

```
LIST

10 PRINT CHR$(147);CHR$(5)
20 POKE 53280,5:POKE 53281,0
30 REM FOR N=1 TO 1000
40 COL=6:      REM COL=RND(0)*15
50 X=20:       REM X=INT(RND(0)*40)
60 Y=15:       REM Y=INT(RND(0)*23)
70 POKE 1024+Y*40+X,90+RND(1)*2:POKE 552
96+Y*40+X,COL
80 REM NEXT N
READY.
```

Qui sopra è riportato il programma grafico di pag. 53; esso è stato modificato in modo che le variabili casuali alle righe da 40 a 60 siano sostituite da valori fissati. Le righe originali sono ancora memorizzate, ma sono disabilitate dai REM; anche il ciclo tra le righe 30 e 80 è mascherato da un paio di REM, in modo che il programma esegua una sola passata.

Se il programma funziona, potete controllare se esso ha eseguito o meno ciò che vi aspettate, e, se non funziona, è comunque più facile ora risalire alla causa del problema. Potete utilizzare questa tecnica con ogni programma che usi delle variabili: sostituendo ogni variabile con una costante potrete controllare il risultato che vi aspettate con quello prodotto dal programma.

Non dimenticatevi che il tasto STOP può essere molto utile per sapere quanta parte di un programma è già stata eseguita. Se eseguite un programma che sembra non produrre alcun risultato, o che si blocca ad un certo punto, il tasto STOP vi indicherà dove si trova il punto morto. Se, poi, visualizzate il testo del programma, sarete quasi sempre in grado di identificare il problema all'interno della riga indicata da STOP e di correggerla in modo che il programma funzioni correttamente.

Come recuperare dei programmi cancellati

Infine, se scrivete molti programmi, vi capiterà, prima o poi, di perdere il testo di un programma perché avete immesso NEW prima di salvarlo su nastro o su disco, anche se non avete ancora immesso nessuna nuova riga di programma. La schermata seguente vi mostra una breve sequenza di comandi immediati per il recupero dei programmi (per CLR si intende SHIFT+CLR):

ANNULLARE NEW

```
POKE 2050,8
SYS 42291
POKE 45,PEEK (174)
POKE 46,PEEK (175)
POKE 47,PEEK (174)
POKE 48,PEEK (175)
POKE 49,PEEK (174)
POKE 50,PEEK (175)
CLR
LIST
```

Questo procedimento funziona perché, quando immettete NEW, il BASIC non cancella veramente il listato del programma dalla memoria. Tutto ciò che succede è che i numerosi puntatori che dicono al BASIC dove il programma comincia e dove memorizzare le sue variabili sono inizializzati con dei valori che fanno supporre al BASIC che non ci sia nessun programma in memoria. Il vostro vecchio programma, in realtà, rimane nella RAM fino a che non ne immettete uno nuovo. Le prime due righe della schermata ricercano in memoria la fine del programma cancellato; questo valore è memorizzato nelle locazioni 174 e 175. Il resto dei comandi ricopia questo indirizzo nei puntatori dai quali era stato cancellato quando avete immesso il comando NEW. I puntatori che si trovano alle locazioni 45/46 e 47/48 sono usati per dire al BASIC dove comincia la lista delle variabili del programma; le locazioni 49/50 indicano la posizione di memoria in cui termina il listato.

COME CONSERVARE I VOSTRI PROGRAMMI

Qualunque testo viene conservato nella memoria del Commodore solo per il tempo che il computer rimane acceso. Quando lo spegnete i vostri programmi scompaiono. Non potete, ovviamente, riscrivere tutti i programmi che volete usare tutte le volte che accendete il vostro Commodore; fortunatamente esso vi offre due modi per salvare i programmi: per mezzo del registratore a cassetta o dell'unità a disco.

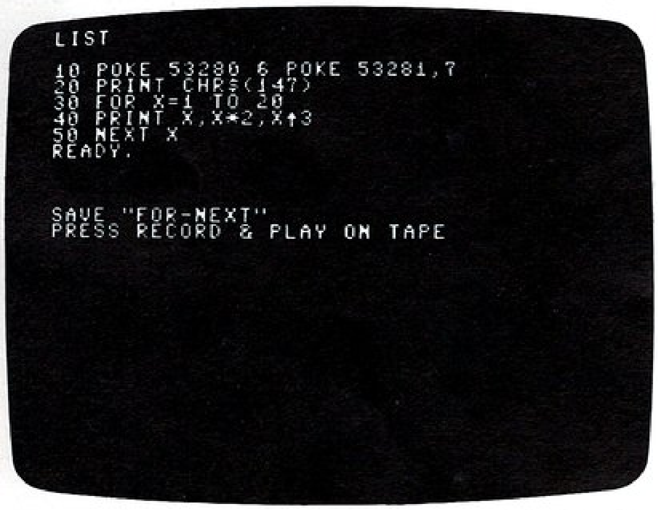
Il registratore a cassetta del Commodore è un apparecchio molto semplice da usare in quanto è progettato appositamente per memorizzare i programmi per il Commodore, e tutti i suoi controlli sono permanentemente predisposti per ricevere il segnale del computer. L'unità a disco vi permette di gestire i vostri dati in modo più sofisticato ma, per l'uso più elementare, esso utilizza gli stessi comandi necessari per il registratore a cassetta.

Comandi per memorizzare i programmi

I programmi vengono registrati su nastro, o su disco, e ricaricati in memoria usando i comandi SAVE e LOAD. Potete provare ad utilizzarli con uno qualsiasi dei programmi presentati in questo libro: immettetene il testo nel computer ed eseguitelo (per essere sicuri che non contenga errori di battitura).

Decidete un nome (lungo al massimo 16 caratteri) da assegnare al programma e scrivete SAVE "nomefile", sostituendo a nomefile il nome che avete scelto. Se state usando un disco, scrivete SAVE "nomefile" ,8. Quando premerete RETURN, l'operazione avrà inizio:

MESSAGGIO DI REGISTRAZIONE SU CASSETTA



Con un registratore, la registrazione comincia quando premete i tasti RECORD e PLAY sul registratore stesso; con una unità a disco, invece, dovete solo immettere il comando SAVE. Quando il computer avrà salvato il programma, apparirà il messaggio READY: il vostro programma dovrebbe essere memorizzato.

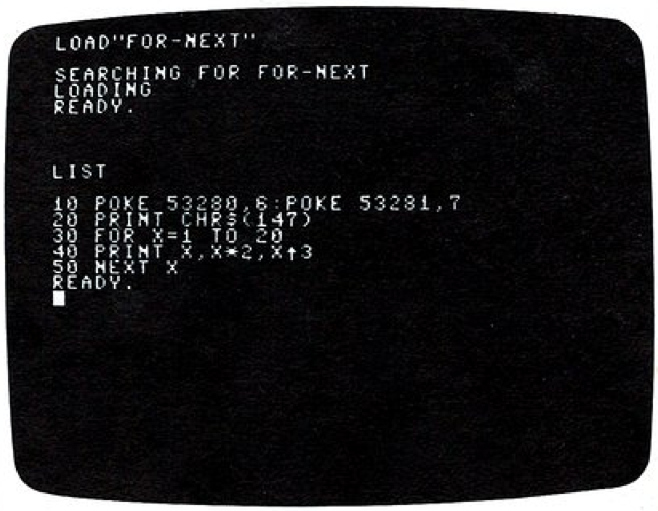
Come controllare nastri e dischi

È sempre bene verificare che il programma che avete appena salvato sia stato registrato correttamente. Per fare ciò immettete il comando VERIFY seguito dal nome del programma (e da ,8 se state usando una unità a disco) e poi premete RETURN. Se state usando il registratore, non appena il computer rileva un programma registrato, ne visualizza il nome sullo schermo; per fare in modo che il computer lo controlli premete il tasto C= (questa operazione non è necessaria con un'unità a disco). Il computer comparerà allora il programma registrato con la versione presente in memoria: se viene riscontrata qualche differenza, è visualizzato il messaggio VERIFY ERROR, per indicarvi che il programma in causa non è stato memorizzato in modo corretto.

Richiamare i programmi in memoria

Se premete ora NEW, potete provare a richiamare il programma in memoria. Scrivete LOAD "nomefile" (se usate il disco ricordatevi di aggiungere ,8). Il programma verrà caricato in memoria pronto per essere eseguito. Con un disco, la testina si sposterà automaticamente nella zona in cui il programma è stato registrato; con un registratore, invece, vi sarà di aiuto usare il contagiri per spostarvi sulla parte opportuna del nastro:

RICHIAMARE UN PROGRAMMA REGISTRATO SU NASTRO



Se immette LOAD "$" ,8 con una unità a disco, il computer caricherà in memoria la directory del disco; il comando LIST vi permetterà allora di visualizzare i nomi di tutti i programmi registrati su disco.

Usare il comando LOAD in un programma

Questo è un modo per far eseguire un programma da un altro programma (operazione conosciuta come "concatenamento"). Se usate il comando LOAD in una riga di programma, il computer caricherà in memoria il programma specificato e quindi lo eseguirà. Con questa tecnica potete collegare assieme un certo numero di programmi; tuttavia, dovete ricordarvi che i valori delle variabili dei programmi precedenti rimangono conservate in memoria.

Ciò non è vero quando usate LOAD come comando diretto, al di fuori di un programma.

GRIGLIA PER GLI SPRITE

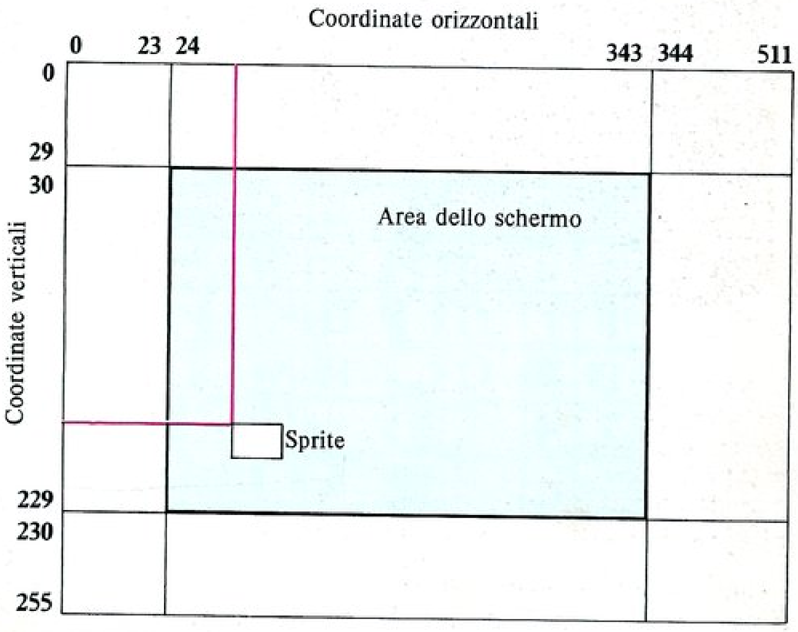
Gli sprite del Commodore sono composti da 21 righe di 24 pixel ciascuna; ogni pixel è controllato da un singolo bit con un'istruzione POKE. Potete usare la griglia seguente per disegnare i vostri sprite e calcolare i valori da memorizzare, che possono essere messi in un'istruzione DATA. Dopo aver disegnato (a matita) il vostro sprite sulla griglia, sommate i valori decimali di tutti i pixel accesi in ogni riga e riportate il totale sulla colonna di destra.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	Valori per l'istruzione DATA
	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	
0																									
1																									
2																									
3																									
4																									
5																									
6																									
7																									
8																									
9																									
10																									
11																									
12																									
13																									
14																									
15																									
16																									
17																									
18																									
19																									
20																									

Posizionare gli sprite

Le coordinate della posizione di uno sprite possono assumere un valore da 0 a 255 (in verticale) e da 0 a 511 (in orizzontale), ma solo una parte di questi valori corrisponde a posizioni realmente visibili sullo schermo. Lo schema a destra mostra queste coordinate: il riquadro centrale rappresenta l'area visibile sullo schermo, che va da 24 a 343 (in orizzontale) e da 30 a 229 (in verticale). Tutto ciò significa che potete facilmente muovere gli sprite fuori e dentro lo schermo.

Dal momento che la posizione verticale varia da 0 a 255, essa può essere controllata da un singolo byte; la posizione orizzontale, d'altra parte ha bisogno di due byte se volete utilizzare tutti i valori da 0 a 511 (=2x256). Se volete che uno sprite continui a muoversi oltre la posizione orizzontale 255, dovete impostare la locazione V+16.



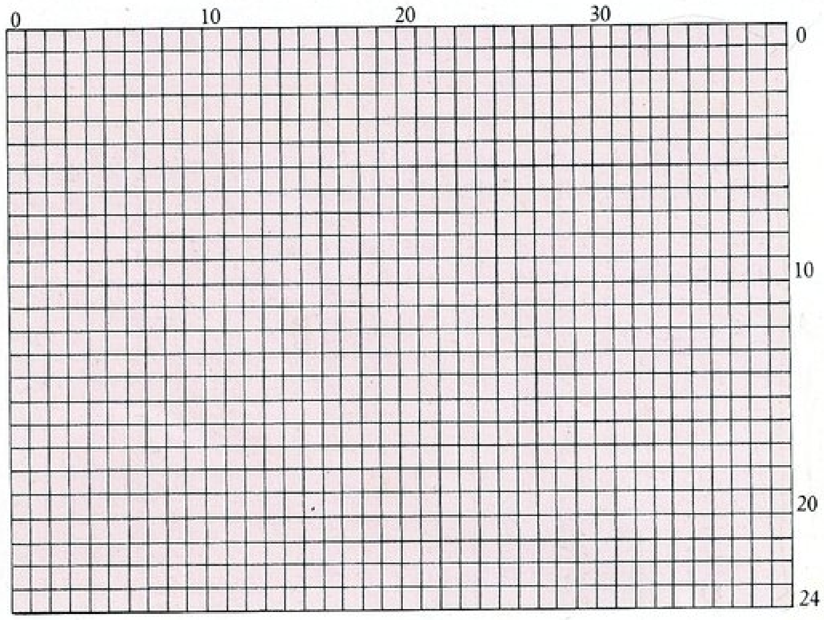
CODICI RELATIVI ALLA MEMORIA VIDEO

In bassa risoluzione si può accedere a ogni punto dello schermo del Commodore con l'istruzione POKE; ciò permette di visualizzare un carattere in un colore qualsiasi in una qualsiasi posizione dello schermo. Sono necessarie, a questo scopo, due istruzioni POKE: una per il carattere da visualizzare in quella posizione, e l'altra per il colore. La griglia seguente aiuta nell'individuare le locazioni delle memorie video e colore.

Memoria video e memoria colore

Per individuare le due locazioni di memoria corrispondenti a una posizione dello schermo, dovete, per prima cosa, determinare le locazioni relative alla posizione verticale; si possono, poi, determinare le locazioni delle memorie video e colore sommando a tali valori il numero di colonna. Determinate queste locazioni, potete memorizzarvi i codici relativi a un carattere ed al suo colore.

1024	55296
1064	55336
1104	55376
1144	55416
1184	55456
1224	55496
1264	55536
1304	55576
1344	55616
1384	55656
1424	55696
1464	55736
1504	55776
1544	55816
1584	55856
1624	55896
1664	55936
1704	55976
1744	56016
1784	56056
1824	56096
1864	56136
1904	56176
1944	56216
1984	56256



I codici dei caratteri

La tabella seguente riporta i codici relativi alle lettere maiuscole e minuscole della tastiera del Commodore: questi sono i codici che devono essere memorizzati nella memoria video. I codici da 128 a 255 producono un'immagine in reverse dei caratteri mostrati qui sotto.

	0	1	2	3	4	5	6	7	8	9
0	@	A a	B b	C c	D d	E e	F f	G g	H h	I i
10	J j	K k	L l	M m	N n	O o	P p	Q q	R r	S s
20	T t	U u	V v	W w	X x	Y y	Z z	[	£	]
30	↑	←	SPAZIO	!	"	#	$	%	&	'
40	(	)	★	+	,	–	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?		♠ A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	● Q	R	♥ S	T	U	V	W	♣ X	Y
90	♦ Z				π		SPAZIO			
100										
110										
120			✓							

L'INSIEME DEI CODICI ASCII

L'insieme dei codici ASCII costituisce una sequenza di caratteri e funzioni a cui si può accedere mediante il comando CHR$; si tratta di un codice digitale standard per i caratteri usati dai computer: i codici da 33 a 127 rappresentano gli stessi caratteri su quasi tutti i calcolatori, mentre gli altri codici sono usati in modo diverso da ciascuna macchina. Sul Commodore tali codici controllano un certo insieme di funzioni, quali la predisposizione dei colori e la rappresentazione dei caratteri speciali. Il codice ASCII usa solo 7 bit di ogni byte, lasciando così spazio per un bit di "parità" usato per il rilevamento di errori di trasmissione in linea.

	0	1	2	3	4	5	6	7	8	9
0						Bianco			Disalibilità SHIFT+C=	Abilità SHIFT+C=
10				RETURN	Lettere maiuscole			Cursore in giù	RVS ON	CLR HOME
20	INST/DEL								Rosso	Cursore a destra
30	Verde	Blu	SPAZIO	!	"	#	$	%	&	'
40	(	)	*	+	,	–	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	£	]	↑	←		♠		
100										╲
110	╱			●		♥			╳	○
120	♣		♦				π			Arancio
130				Tasto Funzione 1	Tasto Funzione 2	Tasto Funzione 3	Tasto Funzione 4	Tasto Funzione 5	Tasto Funzione 6	Tasto Funzione 7
140	Tasto Funzione 8	SHIFT + RETURN	Lettere maiuscole		Nero	Cursore in su	RVS OFF	CLR HOME	INST/DEL	Marrone
150	Rosa	Grigio chiaro	Grigio	Verde chiaro	Azzurro	Grigio scuro	Porpora	Corsore a sinistra	Giallo	Ciano
160	SPAZIO									
170										
180										
190				♠						
200						╲	╱			●
210		♥			╳	○	♣		♦	
220			π		SPAZIO					
230										
240										
250						π				

GLOSSARIO

I termini in **grassetto** sono parole chiave del BASIC.

ASCII
American Standard Code for Information Interchange; è il sistema di codifica dei caratteri usato dal Commodore.

BASIC
Beginner's All-purpose Symbolic Instruction Code; è il linguaggio di programmazione ad alto livello più comunemente usato.

Binario
È il sistema di numerazione usato dai computer e basato solo su due cifre: "0" e "1".

Bit
È la cifra binaria: "0" o "1".

Byte
È un insieme di otto bit.

Chip
È un singolo componente contenente un intero circuito elettronico. È chiamato anche "circuito integrato" (IC = integrated circuit).

CHR$
Converte un codice ASCII nel carattere corrispondente.

CPU
Central Processing Unit. È costituita in generale da un solo chip detto "microprocessore", il quale esegue le operazioni aritmetiche e gestisce il resto del computer.

Cursore
È un simbolo lampeggiante che appare sullo schermo per indicare dove apparirà il prossimo carattere.

DATA
Il programma tratta tutti i valori che seguono la parola chiave **DATA** come informazioni che saranno necessarie nel corso del programma stesso. È usato in coppia con **READ**.

Debugging
È l'operazione di eliminazione degli errori (bug) da un programma.

END
Termina il programma (vedere anche **STOP**).

Filename (nome di file)
È il nome assegnato a un programma o a un insieme di dati che è possibile memorizzare su nastro o su disco.

Flowchart
È un diagramma che visualizza tutti i passi necessari per risolvere uno specifico problema.

FOR...NEXT
È un ciclo che ripete una sequenza di istruzioni del programma per un numero determinato di volte.

GOSUB
Fa saltare il programma alla subroutine che comincia al numero di riga che segue questa parola chiave. La subroutine deve sempre terminare con **RETURN**.

GOTO
Fa saltare il programma al numero di riga specificato dopo questa parola chiave.

Hardware
Si indica con questo termine l'insieme dei componenti meccanici ed elettronici del computer, per distinguerli dai programmi (il software) che li utilizzano.

IF...THEN
Forza il computer a compiere un certo insieme di operazioni se la condizione specificata è vera.

INPUT
Quando è usato in un programma, permette al computer di aspettare che dei dati siano immessi da tastiera.

INT
Arrotonda un numero alla sua parte intera.

Interfaccia
È la connessione hardware e software tra il computer e una sua periferica.

Inviluppo
È la variazione dell'intensità di una nota durante la sua esecuzione. È possibile selezionare diversi inviluppo mediante un'istruzione **POKE**.

K
È l'abbreviazione di Kilobyte (= 1024 byte).

LET
Assegna un valore ad una variabile. L'uso di **LET** è opzionale sul Commodore.

LIST
Permette di visualizzare sullo schermo il programma attualmente memorizzato.

LOAD
Trasferisce un programma dal nastro (o dal disco) alla memoria del computer.

Loop
Si tratta di una sequenza di istruzioni del programma che vengono ripetutamente eseguite fino a che non si verifica una specifica condizione.

NEW
Cancella il programma attualmente in memoria.

Namefile
Vedi Filename

ON...GOTO/GOSUB
Fa saltare il programma a uno dei numeri di riga specificato di seguito, a seconda del valore assunto da una variabile di controllo.

PEEK
Legge il valore memorizzato in una specifica locazione di memoria.

POKE
Memorizza un valore in una specifica locazione di memoria.

PRINT
Permette di visualizzare sullo schermo i termini che seguono questa parola chiave.

RAM
Random Access Memory (memoria volatile). È una memoria che perde il suo contenuto quando viene spento il calcolatore (vedere anche ROM).

READ
Preleva le informazioni contenute in una istruzione **DATA**.

REM
Permette di inserire commenti in un programma. Il computer ignorerà tutto ciò che segue la parola chiave **REM** in una riga di programma.

RESTORE
Predispone il computer per la lettura del primo termine di un insieme di istruzioni **DATA**.

RETURN
Termina una subroutine (vedere anche **GOSUB**).

RND
Fornisce un numero casuale all'interno di un intervallo specificato.

ROM
Read Only Memory (memoria non volatile). Questa memoria viene programmata definitivamente dal suo costruttore e può essere solamente letta da chi usa il computer.

SAVE
Salva il programma in memoria su disco o su nastro. Il programma è identificato da un nome.

SID
Sound Interface Device. È il chip usato dal Commodore per produrre i suoni.

Sintassi
È l'insieme delle regole che governano la forma con cui le istruzioni devono essere utilizzate con linguaggio di programmazione.

Software
Con questo termine vengono definiti i programmi per i computer.

Sprite
È un simbolo grafico, definibile mediante delle istruzioni **POKE**, che si può muovere sullo schermo.

SQR
Fornisce la radice quadrata di un numero.

Statement (Istruzione)
Viene definita con questo termine ogni istruzione di un programma. È possibile inserire più di una istruzione per ogni riga di programma.

STEP
Imposta il passo di un ciclo **FOR...NEXT**.

STOP
Termina il programma e visualizza il numero di riga a cui si trova.

Stringa
È una sequenza di caratteri che vengono trattati come un singolo termine: il nome di qualcuno, per esempio.

Subroutine
È una parte del programma che può essere richiamata quando è necessario, per esempio, allo scopo di costruire una particolare immagine o di eseguire ripetutamente una certa sequenza di calcoli.

TAB
Specifica la posizione in cui verrà stampato il testo su una riga video.

Variabile
È un insieme di locazioni di memoria a cui è assegnato un nome e in cui, durante lo svolgimento del programma, possono essere memorizzate delle informazioni.

VERIFY
Controlla che un programma sia stato salvato correttamente su nastro o su disco per mezzo di **SAVE**.

VIC
Video Interface Circuit. È il chip che si occupa del controllo dello schermo.

INDICE ANALITICO

Le principali occorrenze sono riportate in **grassetto.**

Ringraziamenti
La Dorling Kindersley ringrazia Ian Graham per il suo fondamentale contributo a questa serie. Un ringraziamento anche a Philip Freebrey e a Paul Rubert per l'assistenza tecnica, a Fred Gill per aver ricontrollato il testo e a Richard Bird per l'indice analitico.
La Commodore Business Machine (UK) Ltd ha gentilmente fornito le apparecchiature.

SUPERNOVA

Screen Shot

COLLANA DI PROGRAMMAZIONE

Un corso originale e divertente, di concezione completamente nuova, per imparare da soli a programmare il Commodore 64.
Più di 150 fotografie originali di listati di programmi e di programmi in azione, per mostrare sulla pagina esattamente ciò che accade sullo schermo video.
Pieno di suggerimenti sulle tecniche di programmazione, di schemi e tabelle di riferimento, di consigli su come ottenere i risultati migliori dal vostro Commodore 64.

ALCUNI ARGOMENTI

Come installare il computer e come incominciare a lavorare • Dentro il computer • Come controllare la disposizione dei caratteri sul video • Conversare con il computer • Programmare con gli sprite • Tecniche di animazione • Effetti speciali • Come costruire una banca dati

0026441–6

(18137) Lire 18.500